Positional Knowledge is All You Need: Position-induced Transformer (PiT) for Operator Learning

Junfeng Chen [1] **Kailiang Wu** [1 2 3]

Abstract

Operator learning for Partial Differential Equations (PDEs) is rapidly emerging as a promising approach for surrogate modeling of intricate systems. Transformers with the self-attention mechanism—a powerful tool originally designed for natural language processing—have recently been adapted for operator learning. However, they confront challenges, including high computational demands and limited interpretability. This raises a critical question: *Is there a more efficient attention mechanism for Transformer-based operator learning?* This paper proposes the Position-induced Transformer (PiT), built on an innovative position-attention mechanism, which demonstrates significant advantages over the classical self-attention in operator learning. Position-attention draws inspiration from numerical methods for PDEs. Different from self-attention, position-attention is induced by only the spatial interrelations of sampling positions for input functions of the operators, and does not rely on the input function values themselves, thereby greatly boosting efficiency. PiT exhibits superior performance over current state-of-the-art neural operators in a variety of complex operator learning tasks across diverse PDE benchmarks. Additionally, PiT possesses an enhanced discretization convergence feature, compared to the widely-used Fourier neural operator.

[1]Department of Mathematics, Southern University of Science and Technology, Shenzhen 518055, China [2]Shenzhen International Center for Mathematics, Southern University of Science and Technology, Shenzhen 518055, China [3]National Center for Applied Mathematics Shenzhen (NCAMS), Shenzhen 518055, China. Correspondence to: Kailiang Wu <wukl@sustech.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

1. Introduction

Partial Differential Equations (PDEs) are essential in modeling a vast array of phenomena across various fields including physics, engineering, biology, and finance. They are the foundation for predicting and understanding many complex dynamics in natural and engineered systems. Over the past century, traditional numerical methods, such as finite element, finite difference, and spectral methods, have been well established for solving many PDEs. However, these methods often face challenges for complex nonlinear systems with complicated geometries or in high dimensions.

The advent of machine learning has shifted the paradigm in addressing these challenges in PDEs. Key developments include but are not limited to PINN (Raissi et al., 2019), Deep-Galerkin (Sirignano & Spiliopoulos, 2018), and Deep-Ritz (Yu et al., 2018). These methods are typically solution-learners, i.e., learning a solution of PDEs. They closely resemble traditional approaches such as finite elements, replacing local basis functions with neural networks. While advantageous for high-dimensional problems and complex geometries, solution-learners are usually limited to a single instance, i.e., solving one solution of the PDEs with a fixed initial/boundary condition. To get solution for every new condition, retraining a new neural network is required, which can be very costly.

Solution-learners usually focus on a given PDE. However, for many complex systems the governing equations remain unclear due to uncertain mechanisms, yet identifying underlying PDEs is very challenging without sufficient domain knowledge. Data-driven approaches have emerged as a powerful tool for discovering unknown PDEs or surrogate modeling of known yet complex PDEs. Early strategies include sparse-promoting regression (Rudy et al., 2017; Schaeffer, 2017); however, even after learning the underlying PDEs, numerical solving is still necessary to obtain their solutions. Other techniques, such as PDE-Net (Long et al., 2018; 2019), can learn both the underlying PDE and its solution dynamics.

Another data-driven approach is Flow Map Learning (FML) (Qin et al., 2019; Wu & Xiu, 2020; Chen et al., 2022b; Churchill & Xiu, 2023). Unlike solution-learners, FML is an

operator-learner, which approximates the evolution operator of time-dependent systems with varying initial conditions. Once learned, the flow map or evolution operator can be recursively utilized to predict long-term solution behaviors of the equations for any new initial condition without the need for retraining.

Recently, more versatile operator-learners have been systematically developed for learning mappings between infinite-dimensional function spaces. Existing frameworks include, but are not limited to, the neural operators (Anandkumar et al., 2020; Li et al., 2020; 2021; Kovachki et al., 2023), DeepONets (Lu et al., 2021b; Jin et al., 2022; Lanthaler et al., 2023), principal component analysis-based methods (Bhattacharya et al., 2021), and attention-based methods (Cao, 2021; Kissas et al., 2022; Hao et al., 2023), etc. These operator-learners are applicable to learn the solution operators of parametric PDEs, including mappings from initial conditions to solutions at specific future times, or from boundary conditions, source terms, and model parameters to steady-state solutions.

Transformers, a powerful tool initially designed for natural language processing (Vaswani et al., 2017), have also been adapted for learning operators in PDEs, e.g., Liu et al. (2022); Hao et al. (2023); Xiao et al. (2023). The core of Transformers is the self-attention mechanism. However, conventional self-attention lacks positional awareness, which is found crucial in natural language processing (Vaswani et al., 2017; Shaw et al., 2018) and graph representation (Dwivedi & Bresson, 2020), thus sparking significant research interest (Dai et al., 2019; Dufter et al., 2022). In PDE operator learning, there exist few studies on integrating positional knowledge with self-attention. Cao (2021); Lee (2022) concatenate the coordinates of sampling points with input function values, while Li et al. (2022b) adopt the rotary position embedding technique (Su et al., 2024) to enhance self-attention. Self-attention in operator learning is content-based and relies heavily on the values of input functions. This necessitates distinct attention calculations for each training batch instance, resulting in significant memory usage and high computational costs, especially when compared to the neural operators in Kovachki et al. (2023). This raises critical questions: *Is self-attention indispensable for Transformer-based operator learning? What key positional information is necessary, and how can it be efficiently encoded into Transformer-based neural operators?*

To overcome the challenges of self-attention in operator learning, we propose a novel attention mechanism, termed *position-attention*, from a numerical mathematics perspective. This mechanism is induced by only spatial relations without relying on input function values, marking a significant difference from classical content-based self-attention. Position-attention greatly enhances computational efficiency and effectively integrates positional knowledge. It also resonates with the principles of numerically solving PDEs, offering an interpretable approach to operator learning. Building upon position-attention and its variants, we develop a novel deep learning architecture, termed *Position-induced Transformer* (PiT), for operator learning. Compared to current state-of-the-art neural operators, PiT exhibits superior performance across various benchmarks from elliptic to hyperbolic PDEs, even in challenging cases where the solutions contain discontinuities. Like many neural operators (Azizzadenesheli et al., 2024), PiT features a remarkable discretization convergence property (also called discretization/mesh invariance in the literature (Kovachki et al., 2023; Li et al., 2021)), enabling effective generalization to new meshes which are unseen during training.

The main contributions of this work include:

- We find the importance of positional knowledge, specifically the spatial interrelations of the nodal points where the input functions are sampled, in operator learning. We propose the novel position-attention mechanism and its two variants to effectively incorporate such positional knowledge. Compared to self-attention, position-attention is interpretable from a numerical mathematics perspective and is more efficient for operator learning.
- Based on position-attention and its two variants, we construct PiT, a lightweight Transformer whose training time scales only sub-linearly with the sampling mesh resolution of input/output functions. Moreover, PiT is discretization-convergent, offering consistent and convergent predictions as the testing meshes are refined.
- We conduct numerical experiments on various PDE benchmarks, showcasing the remarkable performance of PiT, and demonstrate its greater robustness in discretization convergence (with 48% smaller prediction error for the Darcy2D benchmark) compared to the Fourier neural operator (FNO). Our code is accessible at `github.com/junfeng-chen/position_induced_transformer`.

2. Approach

2.1. Preliminaries

Operator Learning. Consider generic parametric PDEs:

$$\mathcal{L}_a u = f, \tag{1}$$

where $a \in \mathcal{A}(\Omega_a; \mathbb{R}^{d_a}), u \in \mathcal{U}(\Omega_u; \mathbb{R}^{d_u})$ are functions defined on the bounded domains Ω_a and Ω_u, respectively; $\mathcal{L}_a : \mathcal{U} \to \mathcal{F}$ is a partial differential operator; $f \in \mathcal{F}$; $\mathcal{A}$ and $\mathcal{U}$ are Banach spaces of functions over Ω_a and Ω_u, respectively. As in Li et al. (2021); Anandkumar et al.

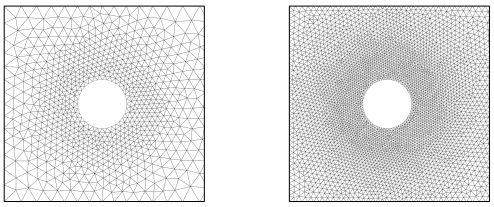

Figure 1. Discretization convergence test for neural operators.

(2020); Li et al. (2020); Kovachki et al. (2023), we assume $\Omega_a = \Omega_u = \Omega \subset \mathbb{R}^d$ in this paper. Denote the operator that maps a to u by Φ, which can be the evolution operator that maps the initial condition to the solution at a specific future time, or the solution operator that maps the source term or model parameters to the steady-state solution. Operator learning aims to construct a neural operator Φ_θ, as surrogate model of Φ, from sampling data pairs $\{a^j, u^j\}_{j=1}^J$. The data are usually sampled on two (possibly different) meshes $X_a = \{x_i\}_{i=1}^{N_a} \subset \Omega$ and $X_u = \{\hat{x}_i\}_{i=1}^{N_u} \subset \Omega$:

$$a^j = \left\{a^j(x_i)\right\}_{i=1}^{N_a}, \quad u^j = \left\{u^j(\hat{x}_i)\right\}_{i=1}^{N_u}, \quad j = 1, 2, ..., J.$$

Assume that the input data $\{a^j\}_{j=1}^J$ are drawn from a probability measure $\mu_{\mathcal{A}}$ supported on $\mathcal{A}$, and the sampling points X_a are i.i.d. drawn from a measure μ_Ω on Ω, denoted as $X_a \sim \mu_\Omega$. After training the parameters θ on the data pairs $\{a^j, u^j\}_{j=1}^J$, we expect that the trained neural operator $\Phi_{\boldsymbol{\theta}}$ exhibits small generalization error defined as

$$\mathbb{E}_{a\sim\mu_{\mathcal{A}}}\left(\|u - \Phi_{\boldsymbol{\theta}}(a|_X)\|_{\mathcal{U}}^2\right), \quad \forall X \sim \mu_\Omega, \tag{2}$$

where the norm $\|\cdot\|_{\mathcal{U}}$ is in practice replaced with a vector norm of the output function values queried on a new mesh $X_{\texttt{new}}$. The formulation (2) indicates that the learned operator $\Phi_{\boldsymbol{\theta}}$ can accept any mesh points in the domain of a and predict u at any queried mesh $X_{\texttt{new}}$.

It is often desirable to achieve a reliable neural operator that is trained with merely inexpensive data on a coarse mesh and yet generalizes well to finer meshes without the need for retraining, as illustrated in Figure 1. In particular, one expects consistent and convergent predictions as the testing meshes are refined (Azizzadenesheli et al., 2024). A common method for assessing this is the *zero-shot super-resolution* evaluation.

A neural operator typically adopts an Encoder-Processor-Decoder architecture:

$$\Phi_\theta = \text{Decoder} \circ \text{LAYER}_L \circ \cdots \circ \text{LAYER}_1 \circ \text{Encoder},$$

where the Encoder lifts the input function from $\mathbb{R}^{d_a}$ to a higher-dimensional feature space $\mathbb{R}^{d_1}$, and the Decoder projects the hidden features from $\mathbb{R}^{d_L}$ to $\mathbb{R}^{d_u}$. The Encoder and Decoder are usually implemented by linear layers, and can include nonlinearities if necessary. In the Processor, let the function $v_\ell : \Omega \to \mathbb{R}^{d_\ell}$ be the continuum of the feature map U_ℓ of LAYER_ℓ. Then, the forward pass $U_{\ell+1} = \text{LAYER}_{\ell+1}(U_\ell)$ approximates the transform

$$v_{\ell+1}(x) = \sigma\left(\int_\Omega \kappa_\ell(x, y, v_\ell(x), v_\ell(y))\, v_\ell(y)\, dy + v_\ell(x) W_\ell\right),$$

where the integral kernel κ_ℓ needs to be parametrized and trained, $W_\ell \in \mathbb{R}^{d_\ell \times d_{\ell+1}}$ is a trainable matrix, and σ is a nonlinear activation function. Various parametrization methods for κ_ℓ have been explored, including but not limited to message passing on graphs (Anandkumar et al., 2020; Li et al., 2020), Fourier transform (Li et al., 2021), and multiwavelet transform (Gupta et al., 2021).

Transformer and Self-attention. Transformers, proposed by Vaswani et al. (2017), are fundamental in natural language processing and form the basis of major advanced language models including GPT and BERT. Their essence lies in the self-attention mechanism. Recently, Kovachki et al. (2023) observed the connections between attention and neural operators, highlighting the potential of Transformers in operator learning. Various specialized and effective Transformers have been developed for operator learning, using Galerkin-type attention (Cao, 2021), hierarchical attention (Liu et al., 2022), cross-attention (Lee, 2022; Li et al., 2022b), mixture of experts (Hao et al., 2023), and orthogonal regularization (Xiao et al., 2023).

Consider $U \in \mathbb{R}^{N_v \times d_\ell}$ as the input sequence comprising N_v elements, each represented by a d_ℓ-dimensional feature vector. Self-attention can be expressed as

$$\text{SelfAtt}(U) = \text{Softmax}\left(\frac{UW^Q(UW^K)^T}{\sqrt{d_{\ell+1}}}\right) UW^V, \tag{3}$$

where $W^Q, W^K, W^V \in \mathbb{R}^{d_\ell \times d_{\ell+1}}$ are trainable matrices. Self-attention is content-based, and it heavily depends on input function values in operator learning. This demands separate attention computations for each training batch instance, leading to intensive memory and computational costs, compared to the neural operators in Kovachki et al. (2023).

2.2. Novel Position-attention and Its Variants

We find the positional knowledge, specifically the spatial interrelations of the sampling points, is essential for operator learning. We propose the *position-attention* mechanism and its two variants, which effectively incorporate such positional knowledge. In contrast to classical self-attention, position-attention does not rely on the input function values themselves, thereby greatly boosting efficiency. Furthermore, position-attention is consistent with the changes of mesh resolution and converges as the meshes are refined.

Position-attention. Let $U \in \mathbb{R}^{N_v \times d_\ell}$ be the values of a generic function v sampled on a mesh X_v of N_v nodal

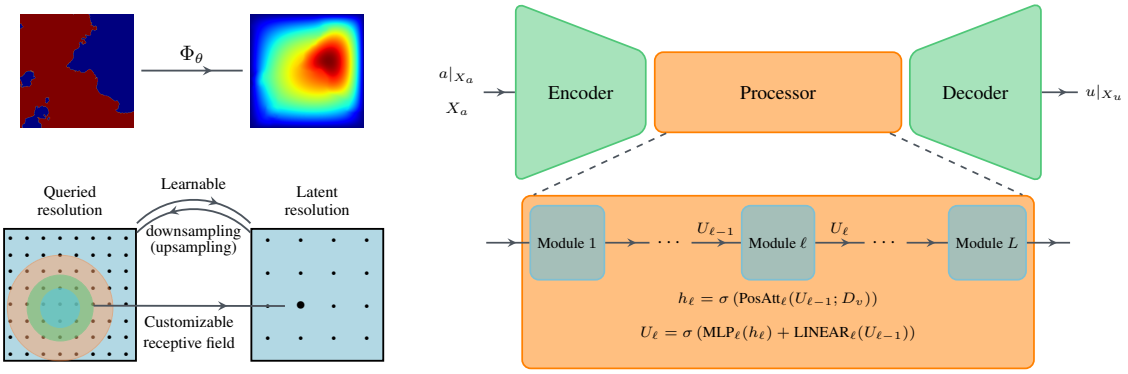

Figure 2. Overview of Position-induced Transformer for operator learning. Top left: A trained neural operator can serve as a surrogate model to specific parametric PDEs. Bottom left: Cross position-attention provides learnable downsampling/unsampling between meshes at different resolutions, and local position-attention supports customizable receptive field. Right: The Encoder-Processor-Decoder architecture of PiT.

points. Define the pairwise-distance matrix $D \in \mathbb{R}^{N_v \times N_v}$:

$$D_{ij} = \|x_i - x_j\|_2^2. \tag{4}$$

Let $\lambda > 0$ and $W^V \in \mathbb{R}^{d_\ell \times d_{\ell+1}}$ be trainable parameters. The position-attention mechanism is defined by

$$\mathrm{PosAtt}(U; D) := \mathrm{Softmax}(-\lambda D) U W^V, \tag{5}$$

which is linear with respect to U. Here, $\mathrm{Softmax}(-\lambda D)U$ can be understood as a global linear convolution, with the kernel weights adjusted according to the relative distances between sampling points. This design is motivated by the concept of *domain of dependence* in PDEs, reflecting how the solution at a point is influenced by the local neighboring information. Specifically, the ith row of the output is

$$\mathrm{PosAtt}(U; D)_i = \sum_{k=1}^{N_v} \frac{\exp(-\lambda D_{ik})}{\sum_{j=1}^{N_v} \exp(-\lambda D_{ij})} (UW^V)_k. \tag{6}$$

Theorem 2.1. *Let $\{X_n\}_{n=1}^{+\infty}$ be a sequence of refined meshes on Ω with $X_n \sim \mu_\Omega$. Denote by D^n the pairwise-distance matrix (4) corresponding to X_n. Assume that $v(x)$ is bounded on Ω, and denote by U^n the function values of v on X_n. As $n \to +\infty$, the position-attention (5) converges to an integral operator: specifically, for any $\varepsilon > 0$,*

$$\lim_{n\to+\infty} \Pr\left\{ \frac{1}{|X_n|} \left\| \mathit{PosAtt}(U^n; D^n) - \mathcal{F}|_{X_n} \right\| \le \varepsilon \right\} = 1,$$

where $|X_n|$ denotes the number of nodal points in X_n,

$$\mathcal{F}(x) := \int_\Omega \kappa^\lambda(x - y)\, v(y) W^V \, d\mu_\Omega(y), \tag{7}$$

and

$$\kappa^\lambda(x - y) = \frac{\exp(-\lambda \|x - y\|_2^2)}{\int_\Omega \exp(-\lambda \|x - y'\|_2^2)\, d\mu_\Omega(y')} \tag{8}$$

is the integral kernel induced by position-attention.

The proof of Theorem 2.1 is put in Appendix B. The fixed measure μ_Ω and the row-wise Softmax normalization play crucial roles in the convergence, which implies that position-attention is discretization-convergent, eliminating the need for nested mesh refinement as required in Kovachki et al. (2023). If one replaces the Softmax normalization with element-wise exponentiation, then κ^λ becomes a Gaussian kernel, which is, however, sensitive to mesh resolution.

The kernel (8) induced by position-attention is independent of the input functions. This is a notable difference from classical self-attention, which is content-based and heavily relies on the input function values themselves. Indeed, position-attention draws inspiration from numerical schemes solving PDEs. For instance, consider the upwind scheme for the advection equation $v_t + s v_x = 0$ with a constant speed s:

$$\begin{aligned} v_j^{n+1} &= v_j^n - \frac{c}{2}(v_{j+1}^n - v_{j-1}^n) + \frac{|c|}{2}(v_{j+1}^n - 2v_j^n + v_{j-1}^n) \\ &=: H_c(v_{j-1}^n, v_j^n, v_{j+1}^n), \end{aligned}$$

where v_j^n is the numerical solution at the jth grid point and time t_n. Here, the operator H_c is discretization-convergent, depending only on a fixed Courant–Friedrichs–Lewy (CFL) number $c := s\Delta t/\Delta x$, and is independent of the input function values $\{v_{j-1}^n, v_j^n, v_{j+1}^n\}$. This scheme can be interpreted as a local linear convolution. Position-attention shares a similar concept but employs a global linear convolution, with the kernel reflecting a stronger dependence on local neighboring regions. Indeed, the value of λ in position-attention is interpretable, as most attention at a queried point x is directed towards points y with the distance to x smaller than $1/\sqrt{\lambda}$; see Appendix D for detailed discussions.

Cross Position-attention. We further propose a novel interpretable variant, *cross position-attention*, for downsampling/upsampling unstructured data. It interpolates U from

a mesh X_1 onto another mesh X_2 by

$$\text{CroPosAtt}(U; D) := \text{Softmax}(-\lambda D_{1\to 2}^T) U W^V, \quad (9)$$

where $D_{1\to 2}$ is the pairwise-distance matrix between X_1 and X_2. As X_1 is refined, cross position-attention also approximates the integral operator defined in equation (7). This property allows us to construct a discretization-convergent Encoder that downsamples the input function values on any mesh X_a to a pre-fixed coarser *latent mesh* X_v, on which the Processor is inexpensive. Analogously, a discretization-convergent Decoder can be constructed to upsample the processed features onto any output mesh X_u.

Remark 2.2. While $X_a, X_v \sim \mu_\Omega$ are important for discretization convergence, we do not require any structure in the output mesh X_u. The output function values can be queried at any point in the domain Ω. The whole model architecture and computational complexity will be detailed in Section 2.3.

Local Position-attention.

The position-attention mechanism naturally captures global dependencies. However, local patterns are often crucial in the solutions of various PDEs, especially those of hyperbolic nature or dominated by convection. To address this, we introduce a local variant of position-attention:

$$\text{LocPosAtt}(U; D)_i = \sum_{D_{ik}\le r_i^2} \frac{\exp(-\lambda D_{ik})}{\sum_{D_{ij}\le r_i^2} \exp(-\lambda D_{ij})} (UW^V)_k. \quad (10)$$

Similar to Theorem 2.1, as the mesh is refined, local position-attention approximates the integral operator

$$\int_{B_{r_i}(x_i)} \kappa^\lambda_{r_{x_i}}(x_i - y)\, u(y) W^V \, \mu_\Omega(dy)$$

with the induced compact kernel

$$k^\lambda_{r_x}(x - y) = \frac{\exp(-\lambda \|x - y\|_2^2)}{\int_{B_{r_x}(x)} \exp(-\lambda \|x - y'\|_2^2)\, d\mu_\Omega(y')}, \quad (11)$$

where $B_{r_x}(x)$, usually termed receptive field, is a ball with radius r_x and center x. We take the radius r_x as a quantile of the row values in the pairwise-distance matrix, adapting the receptive field to the local density of nodal points. This design enables local position-attention to effectively handle functions exhibiting multiscale features. The value of quantile is left as a hyperparameter; see Section 4.6.

2.3. Position-induced Transformer

We now design our novel Transformer architecture, PiT, which is primarily composed of the proposed global, cross, and local position-attention mechanisms for mixing features over the domain Ω. A sketch of the PiT architecture is depicted in Figure 2.

Encoder. The Encoder comprises lifting and downsampling operations using both local and cross position-attention mechanisms:

$$\text{Encoder} = \sigma \circ \text{LocPosAtt}_{\text{in}}(\,\cdot\,; D^T_{a\to v}) \circ \sigma \circ \text{LINEAR},$$

where $D_{a\to v}$ is the pairwise-distance matrix between the input and latent meshes, X_a and X_v; LINEAR refers to a fully connected layer applied row-wisely to the feature matrix. This design allows us to embed the inputs on a coarse mesh into a higher-dimensional feature space, while the local position-attention effectively extracts the local features of the inputs. The dimension d_v of the lifted features is termed the *encoding dimension*, which is an important hyperparameter for the model's expressive capacity.

Processor. The Processor consists of a sequence of global position-attention modules. To address the nonlinearity in general operators, we propose the following module as the building block to construct the Processor:

$$h_\ell = \sigma\left(\text{PosAtt}_\ell(U_{\ell-1}; D_v)\right),$$
$$U_\ell = \sigma\left(\text{MLP}_\ell(h) + \text{LINEAR}_\ell(U_{\ell-1})\right),$$

where D_v is the pairwise-distance matrix of X_v; U_0 is the output of Encoder; MLP_ℓ refers to a multilayer perceptron applied row-wisely to h_ℓ. Throughout our experiments, we stack four global attention modules ($L = 4$) in the Processor with two layers in MLP, and take $d_\ell = d_v$ for all $1 \le \ell \le L$.

Decoder. In the Decoder, we firstly upsample the feature map U_L from the Processor to the queried nodal points X_u, and then apply an MLP row-wisely to project the features back to the range space of the output functions.

$$\text{Decoder} = \text{MLP} \circ \sigma \circ \text{LocPosAtt}_{\text{out}}(\,\cdot\,; D^T_{u\to v}).$$

High Efficiency: Linear Computational Complexity. Due to the kernel matrix multiplication, the forward computation of global position-attention has a quadratic complexity of $O(N_v^2)$. To accelerate large-scale operator learning tasks, we adopt a downsampling-processing-upsampling network architecture. Denote the numbers of nodal points in the meshes X_a, X_v, and X_u by N_a, N_v, and N_u, respectively. We take N_v relatively small for efficiency. The computational complexities in the Encoder and Decoder are $O(N_a N_v d_v + N_a d_v^2)$ and $O(N_u N_v d_v + N_u d_v^2)$, respectively, which are both linear to the numbers of input and output mesh points. This is confirmed by our experiments, where we observe that the training time of PiT scales only sub-linearly with N_a and N_u (see Appendix F.3).

We treat N_v as a hyperparameter of PiT for balancing computational efficiency and information retention on the coarse

latent mesh; see Section 4.6. For training data on structured meshes, we obtain a coarser latent mesh via pooling. For unstructured data, if the distribution is known, one can generate an appropriate latent mesh by sampling; if unknown, one can use farthest point sampling (Zhou et al., 2018) to preserve spatial distribution in the latent mesh.

3. Related Work

Operator Learning. This area is actively researched with numerous related contributions. Chen & Chen (1995) established a universal approximation theorem for approximating nonlinear operators using neural networks. Motivated by this theorem, the DeepONet framework (Lu et al., 2021b), comprising a trunk-net and a branch-net, was proposed for operator learning. The branch-net inputs discretized function values, while the trunk-net inputs coordinates in the domain of the output function. They combine to predict the output function values at specified coordinates. This framework has motivated various extensions, e.g., Jin et al. (2022), Seidman et al. (2022), Lanthaler et al. (2023), Lee et al. (2023), and Patel et al. (2024), etc.

Another pioneering framework is the neural operators based on iterative kernel integration (Anandkumar et al., 2020; Li et al., 2021; Kovachki et al., 2023). Unlike the trunk-branch architecture in DeepONet, this framework typically relies on composing linear integral operators with nonlinear activation functions. The graph neural operator (Anandkumar et al., 2020) leverages message passing to approximate linear kernels in the form $\kappa(x, y)$. FNO (Li et al., 2021) is related to a shift-invariant kernel $\kappa(x - y)$, facilitating operator learning in a frequency domain via discrete Fourier transform. This renders FNO efficient for problems with periodic features. As FNO is limited to uniformly distributed data, various new variants have emerged to handle more complex data structures and geometries, e.g, Geo-FNO (Li et al., 2022a), the non-equispaced Fourier solver (Lin et al., 2022), and the Vandermonde neural operator (Lingsch et al., 2023). Researchers have also developed other related approaches for learning operators in frequency or modal spaces with generalized Fourier projections (Wu & Xiu, 2020), multi-wavelet basis (Gupta et al., 2021), and Laplacian eigenfunctions (Chen et al., 2023).

Transformer-based Neural Operators. Recently, Transformers have been extended to operator learning, including Galerkin and Fourier Transformers (Cao, 2021), HT-net (Liu et al., 2022), MINO (Lee, 2022), OFormer (Li et al., 2022b), GNOT (Hao et al., 2023), and ONO (Xiao et al., 2023). These Transformers are built on self-attention, which relies on the input function values to compute attention weights. This results in distinct attention calculations for each training batch instance, making the Transformer-based neural operators computationally expensive. In contrast, position-attention only rely on the pre-defined pairwise-distance matrix of the sampling points and does not depend on the input function values. This new mechanism notably reduces memory usage and accelerates training.

Cross position-attention, which downsamples the input functions onto coarse latent meshes, also contributes to the high efficiency of PiT. Related ideas include content-based cross-attention used in MINO (Lee, 2022), and bilinear interpolation employed by Galerkin Transformer (Cao, 2021). PiT combines the advantages of both, simultaneously possessing the applicability to irregular point clouds, similar to MINO, and the interpretability, akin to the interpolation in Galerkin Transformer. There are also many other efforts reducing the computational costs of Transformers, such as random feature approximation (Choromanski et al., 2020; Peng et al., 2021), low-rank approximation (Lu et al., 2021a; Xiong et al., 2021), Softmax-free normalization (Cao, 2021), linear cross-attention (Li et al., 2022b; Hao et al., 2023), etc. These techniques may potentially be combined with position-attention to further enhance its efficiency.

Positional/Structural Encoding in Transformers. Transformers, following their success in large language models, have found broad applications in fields such as imaging (Dosovitskiy et al., 2020) and graph modeling (Veličković et al., 2018; Yun et al., 2019). In these models, self-attention is content-based and requires positional encoding. Position information typically falls into two categories: absolute and relative. Vaswani et al. (2017) used sinusoidal functions to encode the absolute positions of words in a sentence. In contrast, Yang et al. (2018) and Guo et al. (2019) focused on the localness of text by adjusting self-attention scores based on word distances. Trainable relative positional encoding was proposed by Shaw et al. (2018); Dai et al. (2019). For graph applications, topological information is as important as position. Structural and positional information in graphs is represented by the graph's Laplacian spectrum (Dwivedi & Bresson, 2020; Kreuzer et al., 2021), shortest-path distance (Ying et al., 2021), and kernel-based sub-graph (Mialon et al., 2021; Chen et al., 2022a), etc.

4. Numerical Experiments

This section presents the experimental results from a variety of PDE benchmarks, demonstrating the superior performance of PiT compared to many other operator learning methods. We also validate the discretization convergence property of PiT in Section 4.3. Section 4.4 presents rigorous comparative studies between self-attention and position-attention. In Section 4.5, we provide some insights on combining self-attention and position-attention. The impacts of hyperparameters are explored in Section 4.6. More experimental results are presented in Appendix F.

4.1. Benchmarks and Baselines

Our tests encompass a diverse range of operator learning benchmarks: **InviscidBurgers** (Lanthaler et al., 2023), **ShockTube** (Lanthaler et al., 2023), **Darcy2D** (Li et al., 2021), **Vorticity** (Li et al., 2021), **Elasticity** (Li et al., 2022a), and **NACA** (Li et al., 2022a). These problems cover elliptic, parabolic, and hyperbolic PDEs, including challenging equations whose solutions exhibit discontinuities. Data for these problems are collected on either structured meshes or irregular point clouds. Due to page limitations, we put the detailed setups of these problems in Appendix A.

We compare PiT with various strong baselines in operator learning: **DeepONet** (Lu et al., 2021b); **shift-DeepONet** (Lanthaler et al., 2023); **FNO** (Li et al., 2021) and **FNO++**, the newest implementation (NeuralOperator, 2023) of FNO using GELU activation and a two-layer MLP after each Fourier layer; **Geo-FNO** (Li et al., 2022a); **Galerkin Transformer** (Cao, 2021); **OFormer** (Li et al., 2022b); **GNOT** (Hao et al., 2023); **ONO** (Xiao et al., 2023). The latter four baselines are all Transformer-based neural operators. Besides the results presented in the following sections, we put more comprehensive comparisons between PiT and the baselines in the appendices; see parameter counts in Appendix E.3; see training speed and memory usage in Appendix E.4.

4.2. Main Results

Table 1 presents the prediction errors of our method and the nine baselines for the six benchmarks. The results for the baselines are directly cited from those original papers, if applicable, and are marked as "−" otherwise. The results of FNO++ are produced using the network hyper-parameters suggested in the references (Li et al., 2021; Lanthaler et al., 2023). Details about the network architectures and training configurations can be found in Appendix E.

In InviscidBurgers and ShockTube, *PiT exhibits excellent performance, comparable to FNO/FNO++, and significant superiority over DeepONet and shift-DeepONet.* Both tasks pose notable challenges due to the discontinuous target functions in the solution operators (see Figures 5 and 6), which are inherently difficult for neural networks to learn. As Lanthaler et al. (2023) pointed out, DeepONet fails to effectively address such difficulties, while shift-DeepONet enhances the performance by incorporating shift-net. PiT overcomes the challenges thanks to its nonlinear Transformer architecture with position-attention. In the InviscidBurgers benchmark, PiT's prediction error is remarkably lower, at just 17% of shift-DeepONet's error and a mere 4.8% of DeepONet's error. In the ShockTube benchmark, PiT's prediction error is about 45% of shift-DeepONet's and 29% of DeepONet's.

Furthermore, in both the Darcy2D and Vorticity benchmarks, *PiT achieves the lowest prediction errors, outperforming all tested baselines.* In the Darcy2D task, PiT's prediction error is only 38% to 57% of those of OFormer, FNO, and Galerkin Transformer. PiT also demonstrates the best performance in the Vorticity benchmark, representing a challenging operator learning task due to data scarcity and the complex patterns of turbulent flow. These results indicate that leveraging spatial interrelations of mesh points is highly beneficial for the attention mechanism in learning these operators.

In the Elasticity and NACA tasks, the PDEs are defined in irregular domains with complex geometries, and the data are sampled on unstructured point clouds for Elasticity and on a deformed mesh for NACA. The complexity of the data and geometry presents significant challenges for operator learning. *Again, PiT, with its position-attention mechanism, exhibits superior accuracy over all the baselines*, including the other four Transformers (OFormer, Galerkin Transformer, GNOT, and ONO) based on self-attention. This suggests that position-attention is crucial, while self-attention might be unnecessary for Transformer-based operator learning.

In addition to its outstanding accuracy, PiT also exhibits high efficiency in terms of training costs. For example, Table 16 displays the training times of PiT with data on various mesh resolutions. These results validate PiT's sub-linear scaling of training time with mesh resolution, consistent with our computational complexity analysis in Section 2.3.

4.3. Discretization Convergence Tests

The Darcy2D dataset, originally collected on a 421^2 Cartesian grid, is downscaled onto a sequence of coarser meshes to serve as reference solutions. This enables the assessment of PiT's discretization convergence via zero-shot super-resolution evaluation. To this end, we train neural operators with data on a coarse mesh. After training, the learned operators are tested on a sequence of refined meshes: 43^2, 61^2, 71^2, 85^2, 106^2, 141^2, 211^2, 421^2, respectively. The testing errors on these meshes are illustrated in Figure 3, where PiT and FNO++ trained on the 43^2 mesh (resp. the 85^2 mesh) are denoted as `PiT43` and `FNO++43` (resp. `PiT85` and `FNO++85`).

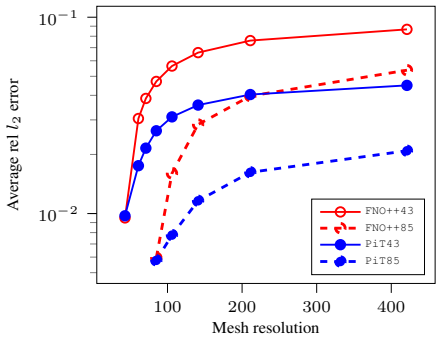

Figure 3. Discretization convergence tests on Darcy2D.

Table 1. Relative errors on the test sets of six benchmarks. The results of InviscidBurgers and ShockTube are reported with the relative l_1 errors. Other benchmarks are evaluated with the relative l_2 errors. The best result of each task is **bolded**, and the second best result is underlined. The data of Darcy2D are represented on a 211×211 uniform grid. The results of Galerkin Transformer and OFormer for Elasticity and NACA are cited from Hao et al. (2023). The results of FNO for Elasticity and NACA are cited from Li et al. (2022a).

	InviscidBurgers	ShockTube	Darcy2D	Vorticity	Elasticity	NACA
DeepONet	0.285	0.0422	–	–	–	–
shift-DeepONet	0.0783	0.0276	–	–	–	–
FNO++	**0.00995**	0.0194	0.00509	0.1315	–	–
FNO	0.0157	0.0156	0.0109	0.1559	0.0508	0.0421
OFormer	–	–	0.0128	0.1755	0.0183	0.0183
Galerkin Transformer	–	–	0.00844	0.1399	0.0201	0.0161
GNOT	–	–	–	0.138	0.00865	0.00757
ONO	–	–	–	0.1195	0.0118	0.0056
Geo-FNO	–	–	–	–	0.0229	0.0138
PiT	0.0136	**0.0122**	**0.00485**	**0.1140**	**0.00649**	**0.00480**

As seen from Figure 3, for operator learning on the 43^2 mesh, FNO++'s prediction error surges from 0.95% to 8.67% as the testing mesh resolution increases to 421^2, while PiT's prediction error rises from 0.97% to only 4.50% (which is 48% lower than that of FNO++). The greater robustness and accuracy of PiT compared to FNO++ are also observed in Figure 3 for the operators trained with 85^2 mesh data. These results demonstrate PiT's superiority over FNO++ in terms of discretization convergence.

4.4. Comparative Ablation Study

We have demonstrated that PiT with position-attention delivers superior performance compared to existing Transformer-based neural operators that utilize self-attention. This suggests that self-attention might not be necessary for operator learning. In this section, we provide a more rigorous ablation study to compare position-attention and vanilla self-attention for operator learning in PDEs. We test two vanilla self-attention models:

SelfAtt A: All PosAtt layers in PiT are replaced with the vanilla self-attention. Three out of six benchmarks encounter an "out of memory" (OOM) issue with a single 24GB RTX-3090 GPU.

SelfAtt B: Only PosAtt layers in Processor are replaced with self-attention, and this avoids the OOM issue.

Table 2 presents the testing errors, parameter counts, and training time. These results validate that PiT is consistently more accurate than Transformers built upon vanilla self-attention, without trade-offs in efficiency.

4.5. Can Self-attention Enhance PiT?

In this section, we aim to answer to such a question: Does combining self-attention and position-attention enhance PiT? To address this, let us consider a Transformer, termed *Self-PiT*, based on a combined attention mechanism:

$$\begin{aligned}&\text{SelfPosAtt}(U; D)\\ &= \text{Softmax}\left(-\lambda D + \frac{UW^Q(UW^K)^T}{\sqrt{d_{\ell+1}}}\right) UW^V. \end{aligned} \tag{12}$$

We have tested Self-PiT on the InviscidBurgers and ShockTube benchmarks, for which the prediction errors are 0.00816 and 0.0179, respectively. By comparing them with the results of PiT in Table 1, we conclude that Self-PiT does not consistently outperform PiT in terms of accuracy, yet it requires more computational complexities.

4.6. Hyperparameter Study

Figure 4 illustrates the impacts of the following three important hyperparameters in PiT.

Quantile in LocPosAtt. A smaller quantile means a more compact receptive field in local position-attention, resulting in a stronger focus on local features. The results in Figure 4 indicate that PiT's performance on ShockTube is sensitive to the quantile in the Encoder but not sensitive to the quantile in the Decoder. Using a small quantile in the Encoder is critical; otherwise, PiT may yield a large prediction error.

Latent Mesh Resolution N_v. This hyperparameter balances computational efficiency and information retention on coarse latent meshes. Choosing a relatively large N_v is crucial to retain essential information in PiT's Processor. Figure 4 shows that the prediction error decreases as N_v increases, as expected. However, this benefit diminishes rapidly as N_v reaches 64. On latent mesh of merely 64 points, PiT is efficient and sufficiently accurate for InviscidBurgers and ShockTube, even though the input data for these tasks is sampled on 1,024 and 2,048 grid points, respectively.

Table 2. Comparisons of position-attention and vanilla self-attention on all benchmark problems. The best result of each task is **bolded**.

	Model	InviscidBurgers	ShockTube	Darcy2D	Vorticity	Elasticity	NACA
Testing errors	SelfAtt A	0.016	0.0259	OOM	OOM	0.0295	OOM
	SelfAtt B	0.0235	0.016	0.0072	0.156	0.169	0.0164
	PiT	**0.0136**	**0.0122**	**0.00485**	**0.114**	**0.00649**	**0.0048**
Parameter counts	SelfAtt A	152,833	152,961	OOM	OOM	9,732,609	OOM
	SelfAtt B	128,263	128,391	444,677	1,776,387	8,684,049	1,774,341
	PiT	**95,503**	**95,631**	**313,613**	**1,252,103**	**6,586,929**	**1,250,061**
Training time second/epoch	SelfAtt A	1.73	5.51	OOM	OOM	**7.13**	OOM
	SelfAtt B	1.47	1.73	15.3	18.7	9.30	21.1
	PiT	**0.938**	**1.04**	**14.7**	**16.3**	7.69	**15.3**

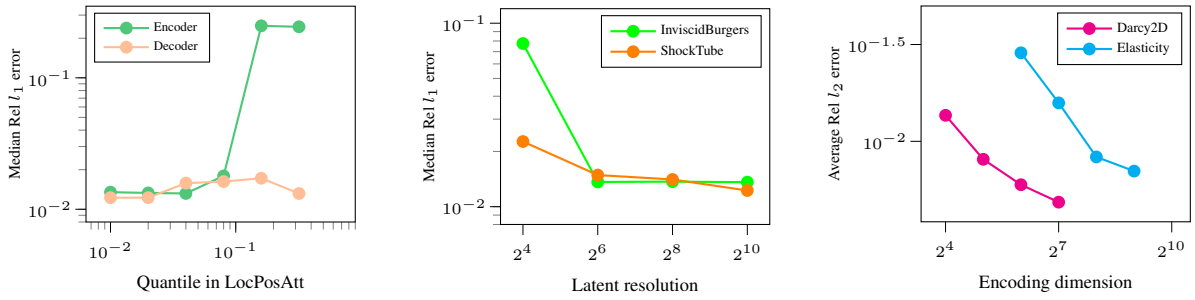

Figure 4. Impacts of the three hyperparameters in PiT.

Encoding Dimension d_v (Network Width). d_v affects the model's expressive capacity. As expected, Figure 4 shows a consistent decrease in relative errors as d_v increases. For Darcy2D, the PiT model with $d_v = 32$ has only $20,000$ trainable parameters, yet its prediction error is merely 0.00808, which is already lower than the errors of FNO, OFormer, and Galerkin Transformer in Table 1. While the latter three methods all have over $2,000,000$ trainable parameters, PiT achieves superior performance with only $20,000$ parameters, demonstrating its parsimonious nature.

5. Conclusions

Inspired by numerical mathematics, we propose position-attention (and two variants) and Position-induced Transformer (PiT) for operator learning in PDEs. PiT exhibits outstanding performance across various PDE benchmarks, surpassing many operator learning baselines. Notably, PiT is discretization-convergent, enabling effective generalization to new meshes with different resolutions. We conclude that the position-attention mechanism is highly efficient for learning nonlinear operators, even in challenging hyperbolic PDEs with discontinuous solutions. Unlike classical self-attention, our position-attention is induced solely by the spatial interrelations of sampling points, without relying on input function values. Position-attention greatly enhances computational efficiency and effectively integrates positional knowledge. Our results demonstrate that position-attention is crucial for operator learning, and *positional knowledge is all you need*.

Impact Statement

PiT emerges as a versatile operator learning framework, applicable to both the surrogate modeling of known parametric PDEs and the data-driven learning of unknown PDEs. It may broadly influence various PDE-related fields such as physics, engineering, biology, and finance, marking an important advancement in AI for science.

Incorporating human insights, which encompass physical and numerical knowledge, is recognized as pivotal in data-driven modeling. Our work not only highlights this integration but also facilitates the development of new operator learning frameworks and the enhancement of existing neural operators. This fosters growth in the realm of scientific machine learning.

One possible negative impact relates to the computational cost of PiT for high-dimensional, large-scale PDEs. To retain essential information of the operators, the latent mesh in PiT may require a large number of nodal points, which notably increases the computational cost. This directs future research endeavors toward further enhancing the current position-attention framework through sparse approximations, low-rank approximations, or Softmax-free variants.

Acknowledgements

This work was partially supported by National Natural Science Foundation of China (Grant No. 92370108) and Shenzhen Science and Technology Program (Grant No. RCJC20221008092757098).

References

Anandkumar, A., Azizzadenesheli, K., Bhattacharya, K., Kovachki, N., Li, Z., Liu, B., and Stuart, A. Neural operator: Graph kernel network for partial differential equations. In *International Conference on Learning Representations*, 2020.

Azizzadenesheli, K., Kovachki, N., Li, Z., Liu-Schiaffini, M., Kossaifi, J., and Anandkumar, A. Neural operators for accelerating scientific simulations and design. *Nature Reviews Physics*, pp. 1–9, 2024.

Bhattacharya, K., Hosseini, B., Kovachki, N. B., and Stuart, A. M. Model reduction and neural networks for parametric pdes. *The SMAI Journal of Computational Mathematics*, 7:121–157, 2021.

Cao, S. Choose a Transformer: Fourier or Galerkin. *Advances in Neural Information Processing Systems*, 34: 24924–24940, 2021.

Chen, D., O'Bray, L., and Borgwardt, K. Structure-aware Transformer for graph representation learning. In *International Conference on Machine Learning*, pp. 3469–3489. PMLR, 2022a.

Chen, G., Liu, X., Li, Y., Meng, Q., and Chen, L. Laplace neural operator for complex geometries. *arXiv preprint arXiv:2302.08166*, 2023.

Chen, T. and Chen, H. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, 6(4):911–917, 1995.

Chen, Z., Churchill, V., Wu, K., and Xiu, D. Deep neural network modeling of unknown partial differential equations in nodal space. *Journal of Computational Physics*, 449:110782, 2022b.

Choromanski, K. M., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J. Q., Mohiuddin, A., Kaiser, L., et al. Rethinking attention with Performers. In *International Conference on Learning Representations*, 2020.

Churchill, V. and Xiu, D. Flow map learning for unknown dynamical systems: overview, implementation, and benchmarks. *Journal of Machine Learning for Modeling and Computing*, 4(2), 2023.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J. G., Le, Q., and Salakhutdinov, R. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2978–2988, 2019.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

Dufter, P., Schmitt, M., and Schütze, H. Position information in Transformers: An overview. *Computational Linguistics*, 48(3):733–763, 2022.

Dwivedi, V. P. and Bresson, X. A generalization of Transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

Guo, M., Zhang, Y., and Liu, T. Gaussian Transformer: a lightweight approach for natural language inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 6489–6496, 2019.

Gupta, G., Xiao, X., and Bogdan, P. Multiwavelet-based operator learning for differential equations. *Advances in Neural Information Processing Systems*, 34:24048–24062, 2021.

Hao, Z., Wang, Z., Su, H., Ying, C., Dong, Y., Liu, S., Cheng, Z., Song, J., and Zhu, J. GNOT: A general neural operator Transformer for operator learning. In *International Conference on Machine Learning*, pp. 12556–12569. PMLR, 2023.

Jin, P., Meng, S., and Lu, L. MIONet: Learning multiple-input operators via tensor product. *SIAM Journal on Scientific Computing*, 44(6):A3490–A3514, 2022.

Kissas, G., Seidman, J. H., Guilhoto, L. F., Preciado, V. M., Pappas, G. J., and Perdikaris, P. Learning operators with coupled attention. *Journal of Machine Learning Research*, 23(1):9636–9698, 2022.

Kovachki, N. B., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A. M., and Anandkumar, A. Neural operator: Learning maps between function spaces with applications to PDEs. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

Kreuzer, D., Beaini, D., Hamilton, W., Létourneau, V., and Tossou, P. Rethinking graph Transformers with spectral attention. *Advances in Neural Information Processing Systems*, 34:21618–21629, 2021.

Lanthaler, S., Molinaro, R., Hadorn, P., and Mishra, S. Nonlinear reconstruction for operator learning of PDEs with discontinuities. In *International Conference on Learning Representations*, 2023.

Lee, J. Y., CHO, S., and Hwang, H. J. HyperDeepONet: learning operator with complex target function space using the limited resources via hypernetwork. In *The Eleventh International Conference on Learning Representations*, 2023.

Lee, S. Mesh-independent operator learning for partial differential equations. In *ICML 2022 2nd AI for Science Workshop*, 2022.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Stuart, A., Bhattacharya, K., and Anandkumar, A. Multipole graph neural operator for parametric partial differential equations. *Advances in Neural Information Processing Systems*, 33:6755–6766, 2020.

Li, Z., Kovachki, N. B., Azizzadenesheli, K., liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.

Li, Z., Huang, D. Z., Liu, B., and Anandkumar, A. Fourier neural operator with learned deformations for PDEs on general geometries. *arXiv preprint arXiv:2207.05209*, 2022a.

Li, Z., Meidani, K., and Farimani, A. B. Transformer for partial differential equations' operator learning. *Transactions on Machine Learning Research*, 2022b.

Lin, H., Wu, L., Xu, Y., Huang, Y., Li, S., Zhao, G., and Li, S. Z. Non-equispaced fourier neural solvers for pdes. *arXiv preprint arXiv:2212.04689*, 2022.

Lingsch, L., Michelis, M., Perera, S. M., Katzschmann, R. K., and Mishra, S. Vandermonde neural operators. *arXiv preprint arXiv:2305.19663*, 2023.

Liu, X., Xu, B., and Zhang, L. Ht-net: Hierarchical Transformer based operator learning model for multiscale PDEs. *arXiv preprint arXiv:2210.10890*, 2022.

Long, Z., Lu, Y., Ma, X., and Dong, B. PDE-Net: Learning PDEs from data. In *International Conference on Machine Learning*, pp. 3208–3216. PMLR, 2018.

Long, Z., Lu, Y., and Dong, B. PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network. *Journal of Computational Physics*, 399:108925, 2019.

Loshchilov, I. and Hutter, F. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2016.

Lu, J., Yao, J., Zhang, J., Zhu, X., Xu, H., Gao, W., Xu, C., Xiang, T., and Zhang, L. SOFT: Softmax-free Transformer with linear complexity. *Advances in Neural Information Processing Systems*, 34:21297–21309, 2021a.

Lu, L., Jin, P., Pang, G., Zhang, Z., and Karniadakis, G. E. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021b.

Mialon, G., Chen, D., Selosse, M., and Mairal, J. Graphit: Encoding graph structure in Transformers. *arXiv preprint arXiv:2106.05667*, 2021.

NeuralOperator. Neuraloperator. https://github.com/neuraloperator/neuraloperator/tree/master, 2023. Accessed in July, 2023.

Patel, D., Ray, D., Abdelmalik, M. R., Hughes, T. J., and Oberai, A. A. Variationally mimetic operator networks. *Computer Methods in Applied Mechanics and Engineering*, 419:116536, 2024.

Peng, H., Pappas, N., Yogatama, D., Schwartz, R., Smith, N., and Kong, L. Random feature attention. In *International Conference on Learning Representations*, 2021.

Qin, T., Wu, K., and Xiu, D. Data driven governing equations approximation using deep neural networks. *Journal of Computational Physics*, 395:620–635, 2019.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Rudy, S. H., Brunton, S. L., Proctor, J. L., and Kutz, J. N. Data-driven discovery of partial differential equations. *Science advances*, 3(4):e1602614, 2017.

Schaeffer, H. Learning partial differential equations via data discovery and sparse optimization. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2197):20160446, 2017.

Seidman, J., Kissas, G., Perdikaris, P., and Pappas, G. J. NOMAD: Nonlinear manifold decoders for operator learning. *Advances in Neural Information Processing Systems*, 35: 5601–5613, 2022.

Shaw, P., Uszkoreit, J., and Vaswani, A. Self-attention with relative position representations. In *Proceedings of NAACL-HLT*, pp. 464–468, 2018.

Sirignano, J. and Spiliopoulos, K. DGM: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.

Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., and Liu, Y. Roformer: Enhanced Transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Wu, K. and Xiu, D. Data-driven deep learning of partial differential equations in modal space. *Journal of Computational Physics*, 408:109307, 2020.

Xiao, Z., Hao, Z., Lin, B., Deng, Z., and Su, H. Improved operator learning by orthogonal attention. *arXiv preprint arXiv:2310.12487*, 2023.

Xiong, Y., Zeng, Z., Chakraborty, R., Tan, M., Fung, G., Li, Y., and Singh, V. Nyströmformer: A Nyström-based algorithm for approximating self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 14138–14148, 2021.

Yang, B., Tu, Z., Wong, D. F., Meng, F., Chao, L. S., and Zhang, T. Modeling localness for self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 4449–4458, 2018.

Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., Shen, Y., and Liu, T.-Y. Do Transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, 34:28877–28888, 2021.

Yu, B. et al. The Deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.

Yun, S., Jeong, M., Kim, R., Kang, J., and Kim, H. J. Graph Transformer networks. *Advances in neural information processing systems*, 32, 2019.

Zhou, Q.-Y., Park, J., and Koltun, V. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.

A. Datasets and Setups of Benchmarks

We briefly present the datasets of the benchmarks considered in this work.

The datasets for InviscidBurgers and ShockTube are obtained from Lanthaler et al. (2023) and are available for download at https://zenodo.org/records/7118642.

The datasets for Darcy2D and Vorticity are obtained from Li et al. (2021) and can be downloaded from https://drive.google.com/drive/folders/1UnbQh2WWc6knEHbLn-ZaXrKUZhp7pjt-.

The datasets of Elasticity and NACA are obtained from Li et al. (2022a) and are accessible for download at https://drive.google.com/drive/folders/1YBuaoTdOSr_qzaow-G-iwvbUI7fiUzu8.

More detailed descriptions about these datasets and setups are provided as follows. *All our codes can be found in the Supplementary Material.*

A.1. Data and Setup for Benchmark 1: Inviscid Burgers

In this benchmark, we consider a nonlinear hyperbolic PDE, namely, the 1D inviscid Burgers' equation (Lanthaler et al., 2023):

$$\begin{aligned} \partial_t u + \partial_x \left(\frac{u^2}{2}\right) = 0, \qquad & (x,t) \in [0,1] \times \mathbb{R}^+, \\ u(\,\cdot\,,0) = \bar{u}(x), \qquad & x \in [0,1], \end{aligned} \tag{13}$$

where the initial condition $\bar{u}(x)$ is sampled from a Gaussian random field. Our objective is to learn the operator that maps various initial conditions to the corresponding entropy solutions at $T = 0.1$. Due to the nonlinear hyperbolic nature of this PDE, its solution can develop discontinuities, even if the initial condition is smooth. This feature makes the task of learning the operator challenging. The solution data at $T = 0.1$ were obtained using a high-order finite volume scheme on a uniform mesh of 1,024 cells (Lanthaler et al., 2023). The full dataset in Lanthaler et al. (2023) comprises 1,024 input-output pairs for training and validation, and 128 pairs for testing. Since we employ a training-testing setup, the validation set with 74 pairs is excluded from our experiments for PiT. In other words, we use only the 950 data pairs as our training set to ensure a fair comparison with the baselines.

A.2. Data and Setup for Benchmark 2: ShockTube

The ShockTube benchmark also involves a nonlinear hyperbolic system of PDEs, whose solutions contain discontinuities. Specifically, we consider the shock tube problem of the 1D compressible Euler equations (Lanthaler et al., 2023):

$$\partial_t \mathbf{U} + \mathbf{F}(\mathbf{U})_x = 0, \quad (x,t) \in [-5,5] \times \mathbb{R}^+, \tag{14}$$

where the conservative vector and flux are respectively given by

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u \\ E \end{pmatrix}, \quad \mathbf{F}(\mathbf{U}) = \begin{pmatrix} \rho u \\ \rho u^2 \\ (E+p)u \end{pmatrix}.$$

Here, ρ, u, and p denote the fluid density, velocity, and pressure, respectively. The total energy, E, consists of the kinetic and internal energies, expressed as $E = \frac{1}{2}\rho u^2 + \frac{p}{\gamma - 1}$, with the constant adiabatic index $\gamma = 1.4$.

The objective is to learn the operator that maps the initial conservative function $\mathbf{U}_0(x)$ to the total energy function $E(x,t)$ at $t = 1.5$. The data for the target total energy function $E(x, t = 1.5)$ are obtained on a uniform grid with 2,048 cells (Lanthaler et al., 2023). The full dataset in Lanthaler et al. (2023) consists of 1,024 and 128 input-output pairs for training and testing, respectively.

A.3. Data and Setup for Benchmark 3: Darcy2D

In the Darcy2D benchmark, we consider an elliptic equation with the Dirichlet boundary condition (Anandkumar et al., 2020; Li et al., 2020; 2021):

$$\begin{aligned} -\nabla \cdot (a \nabla u) = f, \quad & x \in (0,1)^2, \\ u = 0, \quad & x \in \partial(0,1)^2, \end{aligned} \tag{15}$$

where $f = 1$ is the forcing term. Our objective is to learn the operator that maps the permeability $a(x)$ to the pressure field $u(x)$. The dataset comprises 1,024 and 100 input-output pairs for training and testing, respectively. These pairs are obtained by solving the PDE (15) on a 421×421 grid. The permeabilities a are random piecewise constant functions generated according to $a = \psi(\mu)$, where ψ takes 12 for positive realizations of μ and 3 for negative ones, while μ itself is a Gaussian random field with the Neumann boundary condition.

It is worth noting that the input functions of Darcy2D's solution operator, characterized by piecewise constants with jumps at interfaces (see Figure 7), pose difficulties in detecting discontinuities from relatively coarse data.

A.4. Data and Setup for Benchmark 4: Vorticity

This benchmark is related to the two-dimensional incompressible Navier–Stokes equations in the vorticity form (Li et al., 2021):

$$\begin{aligned} \partial_t \omega + \mathbf{u} \cdot \nabla \omega &= \nu \Delta \omega + f, && x \in (0,1)^2, t > 0, \\ \nabla \cdot \mathbf{u} &= 0, && x \in (0,1)^2, t > 0, \\ \omega(\,\cdot\,, t = 0) &= \omega_0, && x \in (0,1)^2, \end{aligned} \tag{16}$$

where $\mathbf{u}(x,t)$ is the velocity field, $\omega = \nabla \times \mathbf{u}$ is the vorticity, $\nu = 10^{-4}$ denotes the viscosity, and $f(x) = 0.1(\sin(2\pi(x_1 + x_2)) + \cos(2\pi(x_1 + x_2)))$ represents a periodic external force. The initial condition, $\omega_0(x)$, is generated by Gaussian random fields. All the data are generated on a 256×256 Cartesian grid and collected with a time lag $\Delta = 1$, then downsampled to a 64×64 grid. The objective is to learn the operator that maps the vorticity snapshots in the time period $t \in [1, 10]$ to the future vorticity snapshots up to $T = 30$. The dataset consists of 1,000 samples for training and 200 samples for testing.

A.5. Data and Setup for Benchmark 5: Elasticity

We consider a hyper-elastic material in a unit cell with a cavity of random shape at the center (Li et al., 2022a). The material is clamped at the bottom side and stretched with a force applied at the upper side. The displacement field for a solid body is governed by the following PDEs (Li et al., 2022a):

$$\rho \frac{\partial^2 \mathbf{u}}{\partial t^2} + \nabla \cdot \boldsymbol{\sigma} = 0 \tag{17}$$

with ρ being the density, $\mathbf{u}$ the displacement, and $\boldsymbol{\sigma}$ the stress tensor. This system is closed by constitutive models relating the strain and stress tensors.

The objective is to learn the solution operator that maps mesh point locations to the displacement field. The dataset in Li et al. (2022a) consists of 1,000 samples for training and 200 samples for testing. Each sample is represented by a point cloud with 972 nodal points. The locations of these points vary across different samples, as the shapes of the cavities are different. See Li et al. (2022a) for more details.

A.6. Data and Setup for Benchmark 6: NACA

This task is related to the transonic flow over airfoils described by the 2D compressible Euler equations (Li et al., 2022a):

$$\partial_t \mathbf{U} + \nabla \cdot \mathbf{F}(\mathbf{U}) = 0, \tag{18}$$

where the conservative vector and flux are respectively

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho \mathbf{v} \\ E \end{pmatrix}, \quad \mathbf{F}(\mathbf{U}) = \begin{pmatrix} \rho v \\ \rho \mathbf{v} \otimes \mathbf{v} + p\mathbf{I} \\ (E + p)\mathbf{v} \end{pmatrix}.$$

Here, ρ represents the density, $\mathbf{v}$ denotes the velocity field, and p is the pressure. The total energy is given by $E = \frac{1}{2}\rho|\mathbf{v}|^2 + \frac{p}{\gamma - 1}$, with $\gamma = 1.4$ being the constant adiabatic index.

The objective is to learn the operator that maps mesh point locations to the Mach number function defined on these mesh points. The dataset in Li et al. (2022a) consists of 1,000 samples for training and 200 samples for testing. Each sample corresponds to a different airfoil shape and is represented on a C-grid mesh refined near the airfoil's surface. The dataset has been transformed onto a regular 221×51 grid.

B. Proof of Theorem 2.1

In this section, we present the proof Theorem 2.1.

Proof. For any $x \in \Omega$, define

$$\mathcal{G}_n(x) := \frac{1}{|X_n|}\sum_{k=1}^{|X_n|}\exp(-\lambda\|x-x_k\|^2)(UW^V)_k,$$
$$\mathcal{H}_n(x) := \frac{1}{|X_n|}\sum_{k=1}^{|X_n|}\exp(-\lambda\|x-x_k\|^2).$$

Then we have

$$\begin{aligned}\mathrm{PosAtt}(U^n;D^n)_i &= \sum_{k=1}^{|X_n|}\frac{\exp(-\lambda D_{ik})}{\sum_{j=1}^{|X_n|}\exp(-\lambda D_{ij})}(UW^V)_k\\ &= \frac{\frac{1}{|X_n|}\sum_{k=1}^{|X_n|}\exp(-\lambda D_{ik})(UW^V)_k}{\frac{1}{|X_n|}\sum_{j=1}^{|X_n|}\exp(-\lambda D_{ij})}\\ &= \frac{\mathcal{G}_n(x_i)}{\mathcal{H}_n(x_i)} =: \mathcal{F}_n(x_i).\end{aligned}$$

Note that $\mathcal{G}_n(x)$ and $\mathcal{H}_n(x)$ are the Monte–Carlo integration approximations to

$$\mathcal{G}(x) := \int_\Omega \exp(-\lambda\|x-y\|_2^2)v(y)W^V d\mu_\Omega(y)$$

and

$$\mathcal{H}(x) := \int_\Omega \exp(-\lambda\|x-y\|_2^2)d\mu_\Omega(y),$$

respectively. By the strong law of large numbers, we know that

$$\Pr\left\{\lim_{n\to\infty}\mathcal{G}_n(x)=\mathcal{G}(x)\right\}=1,$$

and

$$\Pr\left\{\lim_{n\to\infty}\mathcal{H}_n(x)=\mathcal{H}(x)\right\}=1.$$

It follows that

$$\mathcal{F}_n(x)-\mathcal{F}(x)=\frac{\mathcal{G}_n(x)}{\mathcal{H}_n(x)}-\frac{\mathcal{G}(x)}{\mathcal{H}(x)}\xrightarrow{a.s.}0. \tag{19}$$

Define

$$A_n := \int_\Omega\left|\frac{\mathcal{G}_n(x)}{\mathcal{H}_n(x)}-\frac{\mathcal{G}(x)}{\mathcal{H}(x)}\right|d\mu_\Omega(x)=\int_\Omega|\mathcal{F}_n(x)-\mathcal{F}(x)|\,d\mu_\Omega(x).$$

Then, for any $\varepsilon>0$, we have

$$\Pr\left\{\lim_{n\to\infty}A_n=0\right\}=1,\qquad \lim_{n\to\infty}\Pr\left\{A_n\le\frac{\varepsilon}{2}\right\}=1. \tag{20}$$

Using the Chebychev inequality and the elementary inequality $(p-q)^2\le p^2+q^2$ for non-negative p and q, we have

$$\begin{aligned}\Pr\left\{\left|\frac{1}{|X_n|}\sum_{i=1}^{|X_n|}\left|\mathcal{F}_n(x_i)-\mathcal{F}(x_i)\right|-A_n\right|>\frac{\varepsilon}{2}\right\} &\le \frac{4}{|X_n|\varepsilon^2}\int_\Omega\Big(|\mathcal{F}_n(x)-\mathcal{F}(x)|-A_n\Big)^2 d\mu_\Omega(x)\\ &\le \frac{4}{|X_n|\varepsilon^2}\left(A_n^2+\int_\Omega\big(\mathcal{F}_n(x)-\mathcal{F}(x)\big)^2 d\mu_\Omega(x)\right),\end{aligned}$$

which further yields

$$\Pr\left\{\left|\frac{1}{|X_n|}\sum_{i=1}^{|X_n|}\left|\mathcal{F}_n(x_i)-\mathcal{F}(x_i)\right|-A_n\right|\le\frac{\varepsilon}{2}\right\}\ge 1-\frac{4}{|X_n|\varepsilon^2}\left(A_n^2+\int_\Omega\left(\mathcal{F}_n(x)-\mathcal{F}(x)\right)^2 d\mu_\Omega(x)\right).$$

Because

$$\Pr\left\{\frac{1}{|X_n|}\sum_{i=1}^{|X_n|}\left|\mathcal{F}_n(x_i)-\mathcal{F}(x_i)\right|\le A_n+\frac{\varepsilon}{2}\right\}\ge\Pr\left\{\left|\frac{1}{|X_n|}\sum_{i=1}^{|X_n|}\left|\mathcal{F}_n(x_i)-\mathcal{F}(x_i)\right|-A_n\right|\le\frac{\varepsilon}{2}\right\},$$

we obtain

$$\Pr\left\{\frac{1}{|X_n|}\sum_{i=1}^{|X_n|}\left|\mathcal{F}_n(x_i)-\mathcal{F}(x_i)\right|\le A_n+\frac{\varepsilon}{2}\right\}\ge 1-\frac{4}{|X_n|\varepsilon^2}\left(A_n^2+\int_\Omega\left(\mathcal{F}_n(x)-\mathcal{F}(x)\right)^2 d\mu_\Omega(x)\right). \tag{21}$$

Notice that if

$$\frac{1}{|X_n|}\sum_{i=1}^{|X_n|}\left|\mathcal{F}_n(x_i)-\mathcal{F}(x_i)\right|\le A_n+\frac{\varepsilon}{2}\quad\text{and}\quad A_n\le\frac{\varepsilon}{2},$$

then

$$\frac{1}{|X_n|}\sum_{i=1}^{|X_n|}\left|\mathcal{F}_n(x_i)-\mathcal{F}(x_i)\right|\le\varepsilon.$$

This implies

$$\begin{aligned}\Pr\left\{\frac{1}{|X_n|}\sum_{i=1}^{|X_n|}\left|\mathcal{F}_n(x_i)-\mathcal{F}(x_i)\right|\le\varepsilon\right\}&\ge\Pr\left\{\frac{1}{|X_n|}\sum_{i=1}^{|X_n|}\left|\mathcal{F}_n(x_i)-\mathcal{F}(x_i)\right|\le A_n+\frac{\varepsilon}{2}\ \text{ and }\ A_n\le\frac{\varepsilon}{2}\right\}\\&\ge\Pr\left\{\frac{1}{|X_n|}\sum_{i=1}^{|X_n|}\left|\mathcal{F}_n(x_i)-\mathcal{F}(x_i)\right|\le A_n+\frac{\varepsilon}{2}\right\}+\Pr\left\{A_n\le\frac{\varepsilon}{2}\right\}-1,\end{aligned}$$

where the second step follows from the probability inequality $\Pr(a\cap b)\ge\Pr(a)+\Pr(b)-1$. Combing it with (21), we obtain

$$\Pr\left\{\frac{1}{|X_n|}\sum_{i=1}^{|X_n|}\left|\mathcal{F}_n(x_i)-\mathcal{F}(x_i)\right|\le\varepsilon\right\}\ge\Pr\left\{A_n\le\frac{\varepsilon}{2}\right\}-\frac{4}{|X_n|\varepsilon^2}\left(A_n^2+\int_\Omega\left(\mathcal{F}_n(x)-\mathcal{F}(x)\right)^2 d\mu_\Omega(x)\right).$$

Taking $n\to+\infty$ and using (19)–(20), we obtain

$$1\ge\lim_{n\to+\infty}\Pr\left\{\frac{1}{|X_n|}\sum_{i=1}^{|X_n|}\left|\mathcal{F}_n(x_i)-\mathcal{F}(x_i)\right|\le\varepsilon\right\}\ge 1.$$

Therefore,

$$\lim_{n\to+\infty}\Pr\left\{\frac{1}{|X_n|}\left\|\text{PosAtt}(U^n;D^n)-\mathcal{F}|_{X_n}\right\|\le\varepsilon\right\}=1,$$

for any $\varepsilon>0$. This means the position-attention (5) converges in probability to the integral operator (7). The proof is completed. □

C. Technical Aspects of Position-attention

C.1. Ensuring Non-negativity of λ

Ensuring the non-negativity of λ is crucial for position-attention (as well as the cross and local variants) to be well-defined. There are several methods to maintain the non-negativity of during training, including training PiT via constrained optimization or transforming into a positive function of itself (e.g., λ^2). These methods can result in varying performance of the trained neural operator. While replacing λ with λ^2 is an intuitive and straightforward choice, we have empirically observed higher prediction errors in PiT models constructed this way. Conversely, using $\tan(\lambda)$ and training PiTs under the constraint $0 \leq \lambda < \frac{\pi}{2}$ has proven to be more advantageous in certain benchmark cases. This is demonstrated by the results in Table 3.

Table 3. Prediction errors of PiT with two different methods for ensuring non-negativity of λ.

	InviscidBurgers	ShockTube	Darcy2D	Vorticity	Elasticity	NACA
Replace λ with λ^2	0.0273	**0.0122**	0.0102	**0.1169**	**0.00649**	0.162
Use $\tan(\lambda)$ and constrained optimization	**0.0136**	0.0154	**0.00485**	0.5757949	0.00701	**0.00480**

C.2. Multi-head Implementation for Position-attention

In Transformers, the multi-head technique for self-attention is widely used to enhance performance. In parallel, we can formulate the multi-head implementation for position-attention as:

$$\begin{aligned} \text{MultiHeadPosAtt}(U; D) &= \text{Concat}(\text{head}^1, \ldots, \text{head}^h), \\ \text{where head}^i &= \text{Softmax}(-\lambda^i D) U W^i. \end{aligned} \tag{22}$$

Here, h, which divides d_v evenly, represents the number of heads; $\lambda^i > 0$ and $W^i \in \mathbb{R}^{d_v \times \frac{d_v}{h}}$ are the training parameters for the ith head; the Concat operation in (22) concatenates the outputs of all the heads, which are matrices in $\mathbb{R}^{N \times \frac{d_v}{h}}$, to produce the output of MultiHeadPosAtt in $\mathbb{R}^{N \times d_v}$. This implementation also applies to cross and local position-attention.

The multi-head implementation for position-attention defined above performs h independent convolutions with trainable λ^i. With a larger λ^i, the attention decays more rapidly as the pairwise distance increases, indicating a stronger focus on local features. Therefore, multi-head position-attention is trained to provide a balanced view of both local and global aspects of the underlying operator, thereby enhancing the expressive capacity of PiT.

This is corroborated by the results presented in Table 4. Generally, using 2-head implementation in all position-attention layers leads to lower prediction errors than using single-head. Note that 2-head implementation leads to a total of 4 heads for the local position-attention layers in the Encoder and Decoder, and a total of 8 heads for the global position-attention layers in the Processor. We find using more than 2 heads in all position-attention layers hardly improve the performance of PiT.

Table 4. PiT's prediction errors in the six benchmarks when using different number of heads in position-attention and its variants.

	InviscidBurgers	ShockTube	Darcy2D	Vorticity	Elasticity	NACA
$h = 1$	0.998	0.251	0.00627	**0.1140**	0.00794	0.00553
$h = 2$	**0.0136**	**0.0122**	**0.00485**	0.1169	0.00691	**0.00480**
$h = 4$	0.0162	0.0161	0.00502	0.1205	0.00708	0.00507
$h = 8$	0.0157	0.0158	0.00530	0.1186	**0.00649**	0.00545

D. Interpretability of Position-attention

The position-attention mechanism is designed to be interpretable, drawing inspiration from the numerical methods for PDEs, as it has been claimed in Section 2.2. Position-attention shares a similar concept with the upwind scheme and employs

a linear convolution, with the kernel exhibiting a strong dependence on local neighbouring regions, resonating with the principle of *domain of dependence* in PDEs and numerical methods. This insight greatly supports the interpretablility of our method from a theoretical point of view.

We take Darcy2D as an example, where $\tan(\lambda)$ has been used in position-attention as stated in Appendix C. Table 5 shows the values of $1/\sqrt{\tan(\lambda)}$ of all the attention heads (*i.e.* all the convolutions) in the trained PiT model. These values are indeed interpretable, as most attention at a queried point x is directed towards points y with the distance to x smaller than $1/\sqrt{\tan(\lambda)}$.

Table 5. Values of $1/\sqrt{\tan(\lambda)}$ in each attention head. These values are taken from the trained PiT model for the Darcy2D benchmark. This model adopts a 2-head implementation for all the attention layers.

Layer	Attention	Head 1	Head 2
Encoder	LocPosAtt	0.0483	0.0155
Processor 1	PosAtt	2.44	0.840
Processor 2	PosAtt	0.232	0.827
Processor 3	PosAtt	0.382	0.0752
Processor 4	PosAtt	0.588	0.167
Decoder	LocPosAtt	0.0206	0.0498

E. Details of Numerical Experiments

In this section, we outline the training configurations and the neural network architectures to enable easy reproduction of the reported results by readers. All codes and datasets are available through our GitHub repository.

E.1. Training Configurations

As in Li et al. (2021), our experiments are conducted in a train-test setting. We use the mean of the relative l_2 error to train and evaluate models on Darcy2D, Vorticity, Elasticity, and NACA, as in Li et al. (2021; 2022a). For InviscidBurgers and ShockTube, the mean of the relative l_1 error is used for training, while performance is assessed using the median of the relative l_1 error on the testing set, following Lanthaler et al. (2023). The models are trained using the Adam optimizer and a cosine annealing learning rate scheduler (Loshchilov & Hutter, 2016). The initial learning rate is set to 0.001, and the training lasts for 500 epochs. The batch sizes adopted in the experiments are shown in Table 6. All the experiments are performed on an NVIDIA GTX 3090 GPU card. PiT employs an Encoder-Processor-Decoder architecture. Detailed

Table 6. Batch size.

	InviscidBurgers	ShockTube	Darcy2D	Vorticity	Elasticity	NACA
# Training samples	950	1,024	1,024	1,000	1,000	1,000
Batch size	5	8	8	8	10	8

network architectures for the six benchmark problems are presented in Table 9. We begin by introducing some abbreviations representing the basic computational layers:

- PosAtt(w, h): A global position-attention layer followed by the GELU activation, where w is the encoding dimension and h is the number of attention heads.
- LocPosAtt$(w, h,$ down (or up)$)$: A local position-attention layer followed by the GELU activation, with w as the encoding dimension and h as the number of attention heads. This layer's locality parameter is indicated by a quantile value, defining the compactness of the receptive field. Downsampling or upsampling is integrated with the local position-attention. The quantile value and the latent resolution within PiT are detailed in Table 7 and Table 8, respectively.

- LINEAR$(w, \text{activate})$: A fully connected layer with w neurons. This layer applies pointwisely to feature vectors. It can be optionally activated by the GELU function.
- MLP(w_1, w_2): two stacked Linear layers with respectively w_1 and w_2 neurons. The first layer is activated by the GELU function.

Table 7. Quantile value for each benchmark task. In local position-attention, smaller value means more compact recptive field.

	InviscidBurgers	ShockTube	Darcy2D	Vorticity	Elasticity	NACA
Quantile in Encoder	1%	4%	2%	1%	2%	0.5%
Quantile in Decoder	8%	2%	5%	8%	2%	2%

Table 8. Input and latent mesh resolutions for each task. (*) The data of Elasticity are sampled on irregular point clouds.

	InviscidBurgers	ShockTube	Darcy2D	Vorticity	Elasticity*	NACA
Input resolution	$1,024 \times 1$	$2,048 \times 1$	211×211	64×64	972	221×51
Latent resolution	$1,024 \times 1$	$1,024 \times 1$	32×32	16×16	972	111×26

Table 9. Details of the PiT architectures for all six benchmarks.

	InviscidBurgers	ShockTube	Darcy2D	Vorticity	Elasticity	NACA
Encoder	Linear$(64, \text{activate})$ LocPosAtt$(64, 2, \text{down})$	Linear$(64, \text{activate})$ LocPosAtt$(64, 2, \text{down})$	Linear$(128, \text{activate})$ LocPosAtt$(128, 2, \text{down})$	Linear$(256, \text{activate})$ LocPosAtt$(256, 1, \text{down})$	Linear(512) LocPosAtt$(512, 8)$	Linear$(256, \text{activate})$ LocPosAtt$(256, 2, \text{down})$
Processor	$4 \times [$ PosAtt$(64, 2)$ MLP$(64, 64)$ LINEAR(64) GELU $]$	$4 \times [$ PosAtt$(64, 2)$ MLP$(64, 64)$ LINEAR(64) GELU $]$	$4 \times [$ PosAtt$(128, 2)$ MLP$(128, 128)$ LINEAR(128) GELU $]$	$4 \times [$ PosAtt$(256, 1)$ MLP$(256, 256)$ Linear(256) GELU $]$	$4 \times [$ PosAtt$(512, 8)$ MLP$(512, 512)$ LINEAR(512) GELU $]$	$4 \times [$ PosAtt$(256, 2)$ MLP$(256, 256)$ LINEAR(256) GELU $]$
Decoder	LocPosAtt$(64, 2, \text{up})$ $[$PosAtt$(64, 2)$ MLP$(64, 64)$ LINEAR(64) GELU$]$ MLP(64,1)	LocPosAtt$(64, 2, \text{up})$ $[$PosAtt$(64, 2)$ MLP$(64, 64)$ LINEAR(64) GELU$]$ MLP(64,1)	LocPosAtt$(128, 2, \text{up})$ MLP$(128, 1)$	LocPosAtt$(256, 1, \text{up})$ MLP$(256, 1)$	LocPosAtt$(512, 8)$ MLP$(512, 1)$	LocPosAtt$(256, 2, \text{up})$ MLP$(256, 1)$
# Parameters	$95,503$	$95,631$	$313,613$	$1,252,103$	$6,586,929$	$1,250,061$
Training time (seconds/epoch)	0.938	1.04	14.7	16.3	7.69	15.3

We present the details of FNO++[1] in Table 10. For datasets on regular grids, FNO++ shows outstanding training speed thanks to the fast discrete Fourier transform.

Table 10. Architecture details and training times of FNO++.

	InviscidBurgers	ShockTube	Darcy2D	Vorticity
Modes	19	7	12	12
Width	32	32	32	20
# Parameters	$170,593$	$72,353$	$2,376,449$	$928,661$
Training time (seconds/epoch)	0.434	0.671	5.64	11.5

[1] https://github.com/neuraloperator/neuraloperator/tree/master

E.2. Insights on Hyper-parameter Calibration

Tuning the hyper-parameters (quantile, latent resolution N_v, and encoding dimension d_v) in PiT models is not a difficult process. We recommend beginning with a small quantile value (for instance, 1%) for the Encoder and Decoder, using a coarse latent mesh, and setting the encoding dimension to 64. These initial settings typically allow a PiT model to deliver comparable performance to baseline models for all our tested cases.

Should there be a need for further refinement to attain higher accuracy, we proceed with localized tuning. This involves increasing the encoding dimension d_v first. If the desired accuracy is not met through this adjustment, we then refine the latent resolution N_v and, as a final step, modify the quantile. This stepwise approach helps in efficiently reaching the optimal performance without exhaustive search.

E.3. Comparison of Parameter Counts with Baselines Models

For further benchmark purpose, we provide the parameter count for each of the models in Table 1.

- For the InviscidBurgers and ShockTube benchmarks, we have gathered parameter counts of DeepONet, shift-DeepONet, FNO and FNO++ in Table 11:

Table 11. Parameter counts of PiT and the baseline models in the InviscidBurgers and ShockTube benchmarks. The smallest model in each task is **bolded**, and the second smallest model is underlined.

Model	InviscidBurgers	ShockTube
DeepONet	$618,085$	$3,190,673$
shift-DeepONet	$1,835,297$	$6,047,633$
FNO	$\mathbf{90,593}$	$\mathbf{41,505}$
FNO++	$170,593$	$\underline{72,353}$
PiT	$\underline{95,503}$	$95,631$

- For the Darcy2D and Vorticity benchmarks, we have gathered parameter counts for Galerkin Transformer, OFormer, FNO, and FNO++ in Table 12:

Table 12. Parameter counts of PiT and the baseline models in the Darcy2D and Vorticity benchmarks.

Model	Darcy2D	Vorticity
Galerkin Transformer	$\underline{2.22 \text{ Million}}$	1.56 Million
OFormer	2.51 Million	1.85 Million
FNO	$2,368,001$	$\mathbf{926,517}$
FNO++	$2,376,449$	$\underline{928,661}$
PiT	$\mathbf{313,613}$	$1,252,103$

- For the Elasticity and NACA benchmarks, we have gathered the parameter counts for FNO and Geo-FNO in Table 13. It is worth noting that, for the Elasticity benchmark, we can reduce the encoding dimension of PiT from 512 to 256, yielding a model with only $\mathbf{1,655,053}$ parameters (less than those of FNO and Geo-FNO). While this adjustment increases the testing error of PiT from 0.00649 to 0.00829 (as shown in Table 1 and Figure 4), it still remains lower than the testing errors of all baseline models. This further demonstrates the efficiency and accuracy of PiT.

E.4. Comparison of Training Costs with Baseline Transformer-based Neural Operators

To validate the superior efficiency of our PiT model over other Transformer architectures, we have acquired the following runtime and memory results:

Table 13. Parameter counts of PiT and the baseline models in the Elasticity and NACA benchmarks.

Model	Elasticity	NACA
FNO	$\mathbf{2,368,001}$	$\underline{2,368,001}$
Geo-FNO	$\underline{3,020,963}$	$4,727,329$
PiT	$6,586,929$	$\mathbf{1,250,061}$

- In the Darcy2D benchmark, Cao (2021) reported a training time of 0.61 hours for 100 epochs using the 211×211 dataset, equating to 22 seconds/epoch or 5.83 iterations/second (with 128 iterations/epoch at a batch size of 8), with a Galerkin Transformer comprising 2.22 million parameters. In contrast, our PiT model required only 14.7 seconds/epoch, with a significantly lower parameter count of 0.31 million, using the same dataset and GPU.
- For the Vorticity test case with $\nu = 10^{-4}$ employing $10,000$ training samples, Li et al. (2022b) detailed the training costs for OFomer and the Galerkin Transformer. Our experiments with PiT, adhering to the same batch size and GPU, demonstrated significantly greater efficiency over these two Transformer architectures, as shown in Table 14

Table 14. Iteration per second and memory usage during model training. The best results are **bolded**, and the second best results are underlined.

Model	Iters/sec	Memory (GB)	params #(Million)
Galerkin Transformer	1.79	16.65	$\underline{1.56}$
OFormer	$\underline{1.89}$	$\underline{15.93}$	1.85
PiT	$\mathbf{6.71}$	$\mathbf{4.4}$	$\mathbf{1.25}$

F. Additional Experimental Results

In this section, we present additional experimental results to support our findings and demonstrate the effectiveness of the proposed PiT for complex operator learning tasks.

F.1. Additional Experimental Results on Benchmark 1: InviscidBurgers

In Figure 5, we present some predicted solutions of the inviscid Burgers' equation obtained using the Self-PiT. Due to the nonlinear hyperbolic nature of the PDE, many discontinuities are developed in the solutions, even though the initial conditions are smooth. We observe an excellent agreement between the predicted and reference solutions.

In Table 15, we display the 25% and 75% quantiles of the relative l_1 errors computed over the testing data for PiT compared with the baselines. It is evident that PiT and Self-PiT achieve outstanding results compared to the baselines, although Self-PiT does not consistently outperform PiT.

Table 15. 25% and 75% quantiles of relative l_1 errors ($\times 10^{-2}$) on the testing dataset.

	DeepONet	shift-DeepONet	FNO	FNO++	PiT	Self-PiT
InviscidBurgers	$25.4 - 32.4$	$6.7 - 9.6$	$1.3 - 1.9$	$0.842 - 1.26$	$1.1 - 1.61$	$\mathbf{0.651 - 0.974}$
ShockTube	$3.4 - 5.4$	$2.0 - 3.75$	$1.2 - 2.1$	$1.48 - 2.68$	$\mathbf{0.9 - 1.82}$	$1.4 - 2.64$

F.2. Additional Experimental Results on Benchmark 2: ShockTube

In this operator learning task, the initial density, momentum and energy are step-shaped functions. The 1D compressible Euler equations evolve these initial conditions into various wave structures with discontinuities at $t = 1.5$ (see Figure 6). The results predicted by PiT show good agreement with the reference solutions.

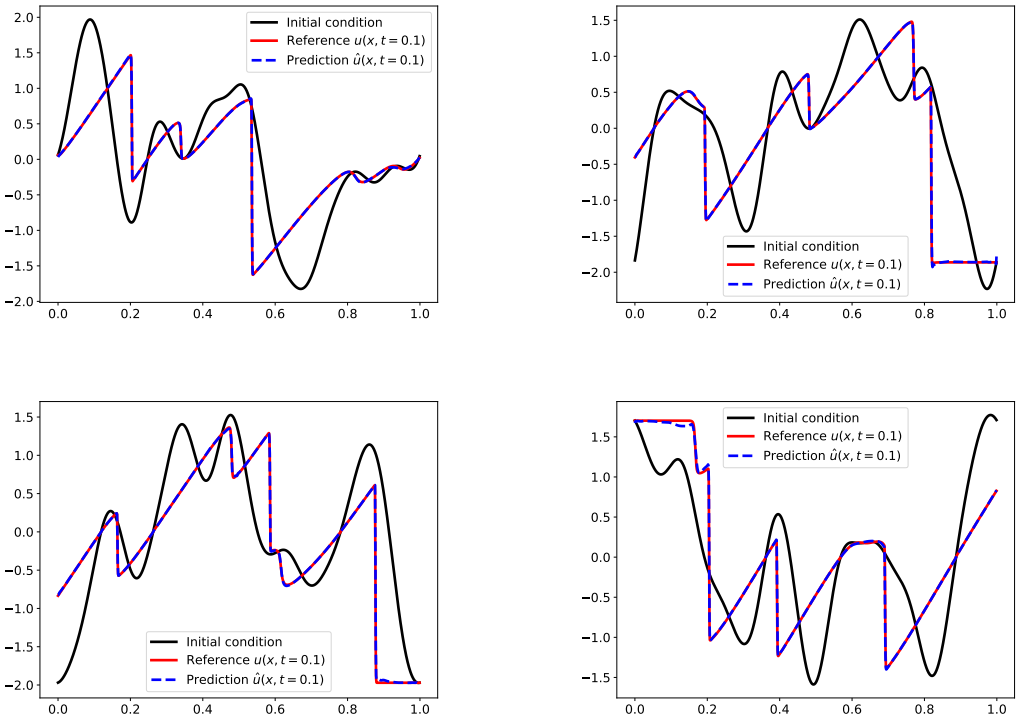

Figure 5. InviscidBurgers: Predictions given by Self-PiT for four different input functions.

In Table 15, we present the 25% and 75% quantiles of the relative l_1 errors over the testing data for PiT in comparison with the baselines. For this benchmark, PiT achieves the lowest prediction error, surpassing all tested baselines.

F.3. Additional Experimental Results on Benchmark 3: Darcy2D

We have shown that PiT demonstrates exceptional efficiency in approximating the solution operator for the Darcy2D benchmark. Remarkably, a PiT trained on downscaled 43^2 data can accurately predict solutions on the full 421^2 grid, achieving only a 4.50% relative error.

The latent resolution of PiT is fixed at 32^2, regardless of the mesh resolution of the input functions. When trained with data at a finer resolution, PiT shows enhanced performance. We document the prediction error and the training cost for different datasets in Table 16. Notably, the training time per epoch scales sub-linearly with the number of mesh points, denoted by N. Furthermore, the prediction error rapidly reduces as N increases. These results highlight PiT's effectiveness in learning large-scale operators and its remarkable discretization-convergent property under zero-shot super-resolution evaluation (see Figure 7).

F.4. More Experimental Results on Benchmark 4: Vorticity

In Figure 8, the growth of prediction error is plotted. We observe approximately an exponential trend, which is normal for data-driven evolution operators. In Figure 9, we present the evolution of the vorticity field. Although Vorticity is a hard task with turbulent flow pattern and scarce data, our method correctly captures the evolution pattern.

F.5. Additional Experimental Results on Benchmark 5: Elasticity

The Elasticity benchmark presents a unique challenge as it comprises samples with cavities of varying shapes. Unlike other tasks that use a fixed grid for all data samples, the sampling points in the Elasticity dataset are body-fitted to the cavities. This feature makes Elasticity distinct. As further demonstrated in Figure 10, PiT effectively captures the stress concentration

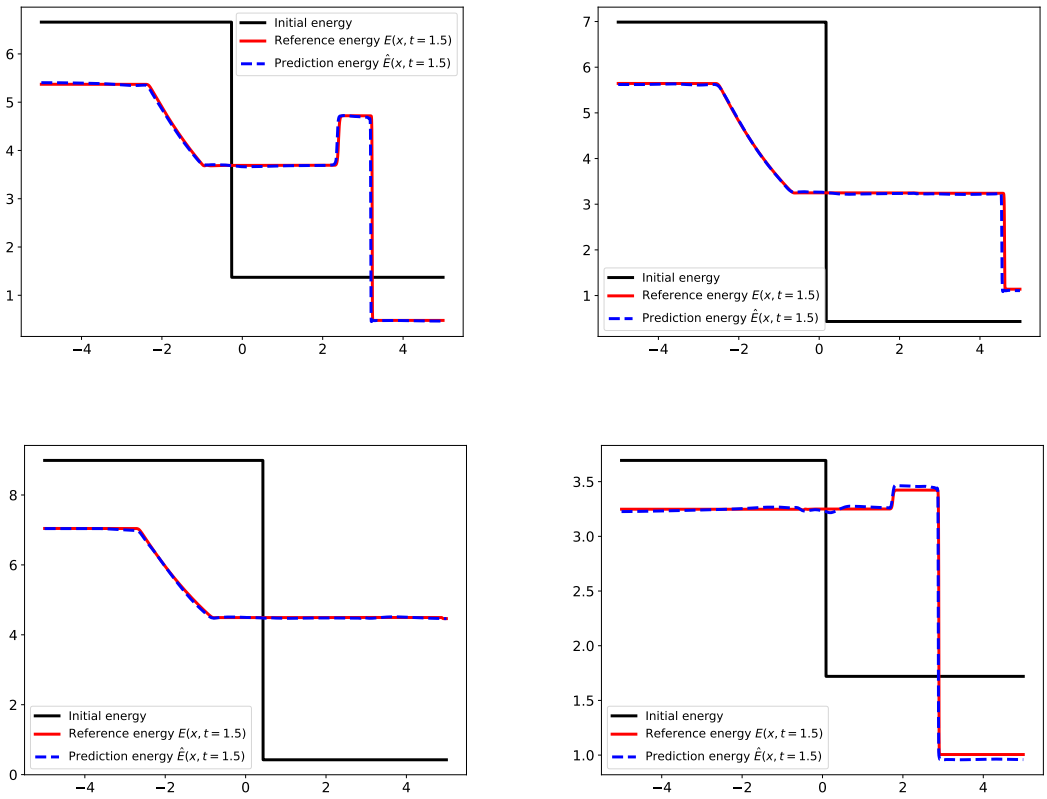

Figure 6. ShockTube: Predictions of the total energy functions at $t = 1.5$ given by PiT for four different input functions, compared to the reference solutions. The initial conditions for density, momentum and energy are all step functions.

Table 16. Darcy2D: Results of discretization-convergent experiments. Four PiT models are trained with data on different mesh resolutions, and then evaluated using the testing data on either the same mesh or the finer 421^2 mesh. The prediction errors are measured by average relative l_2 errors.

Training resolution	43^2	85^2	141^2	211^2
Training time (seconds/epoch)	1.11	2.41	4.37	14.7
Prediction error on training resolution	0.00974	0.00578	0.00558	0.00485
Prediction error on 421×421 resolution	0.0450	0.0209	0.0117	0.00715

resulting from the irregular geometries of the cavities.

We have found that the parametrization of the cavity's shape is crucial for learning the target operator. Each sample in the dataset is characterized by a 42-dimensional vector, which is utilized to parametrize the shape of the cavity. We concatenate this vector with the mesh point coordinates to serve as the input for our model. In other words, the input is a tensor with 44 channels, of which 42 are constants across the domain. To improve the performance, we apply the transformation $g(r) = 5r - 1$ to each channel of the shape parameters, effectively normalizing their values to the interval $[0, 1]$. This normalization ensures that the shape features are comparable to the coordinate features, considering that the problem is defined within the unit square $[0, 1]^2$.

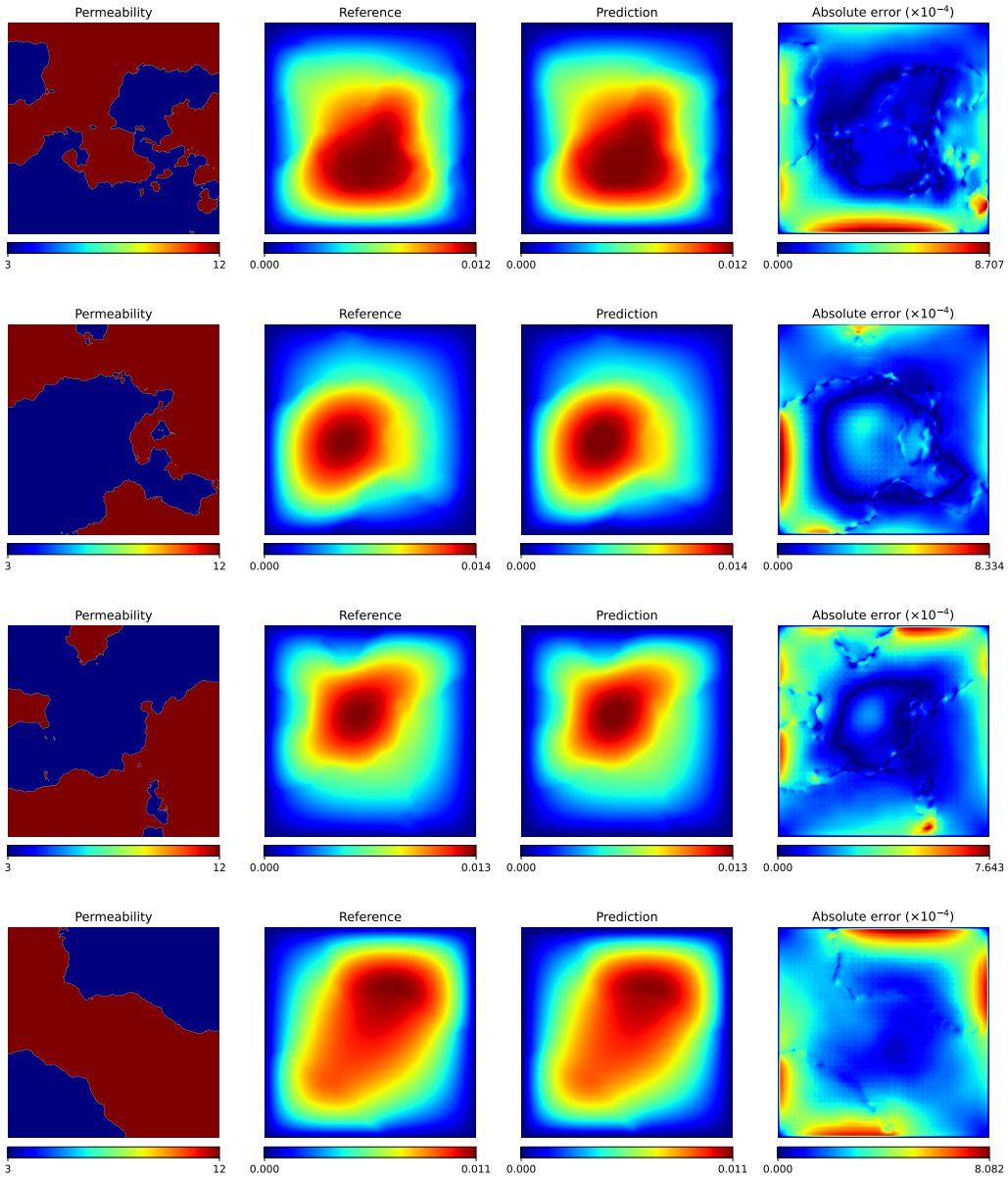

Figure 7. Darcy2D: The input functions (permeability), the referential and predicted pressure fields, and the absolute error (from left to right). The model is trained on 43^2 mesh resolution, and then tested on 421^2 mesh resolution. From top to bottom: four testing examples with different input functions.

F.6. Additional Experimental Results on Benchmark 6: NACA

The data for the NACA benchmark are sampled on C-grid meshes, which are locally refined near the surfaces of the airfoils. The mesh points cover a large domain encompassing the airfoil, where the chord length is set to 1.

Figure 11 displays a close-up view of the Mach field around the airfoils, produced by the learned operator of PiT. One can see that PiT accurately predicts the Mach field and effectively captures the shock wave structures. The predicted solutions are in good agreement with the reference solutions.

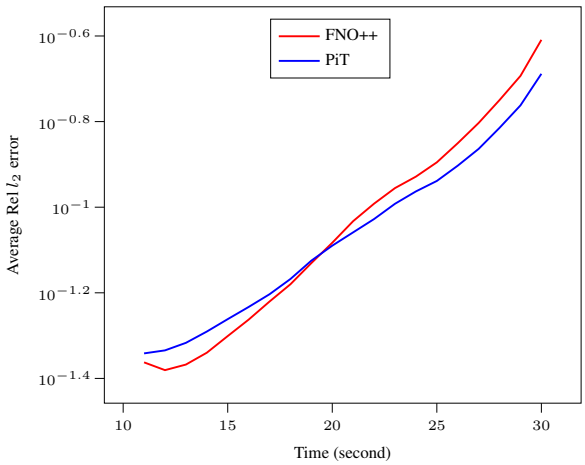

Figure 8. Vorticity: Evolution of prediction errors over time.

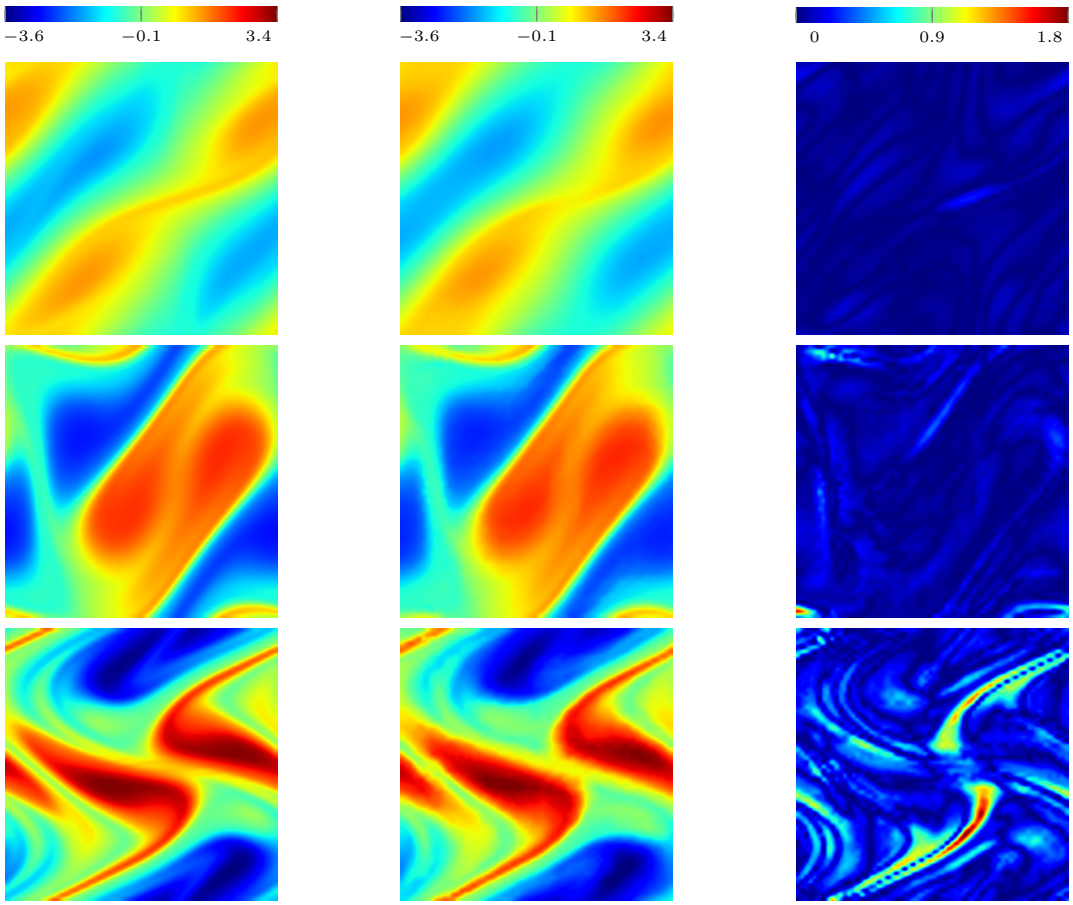

Figure 9. Vorticity benchmark: Evolution of the vorticity field at $t = 11$, 20, and 30 (from top to bottom). Left: reference. Middle: prediction. Right: absolute error.

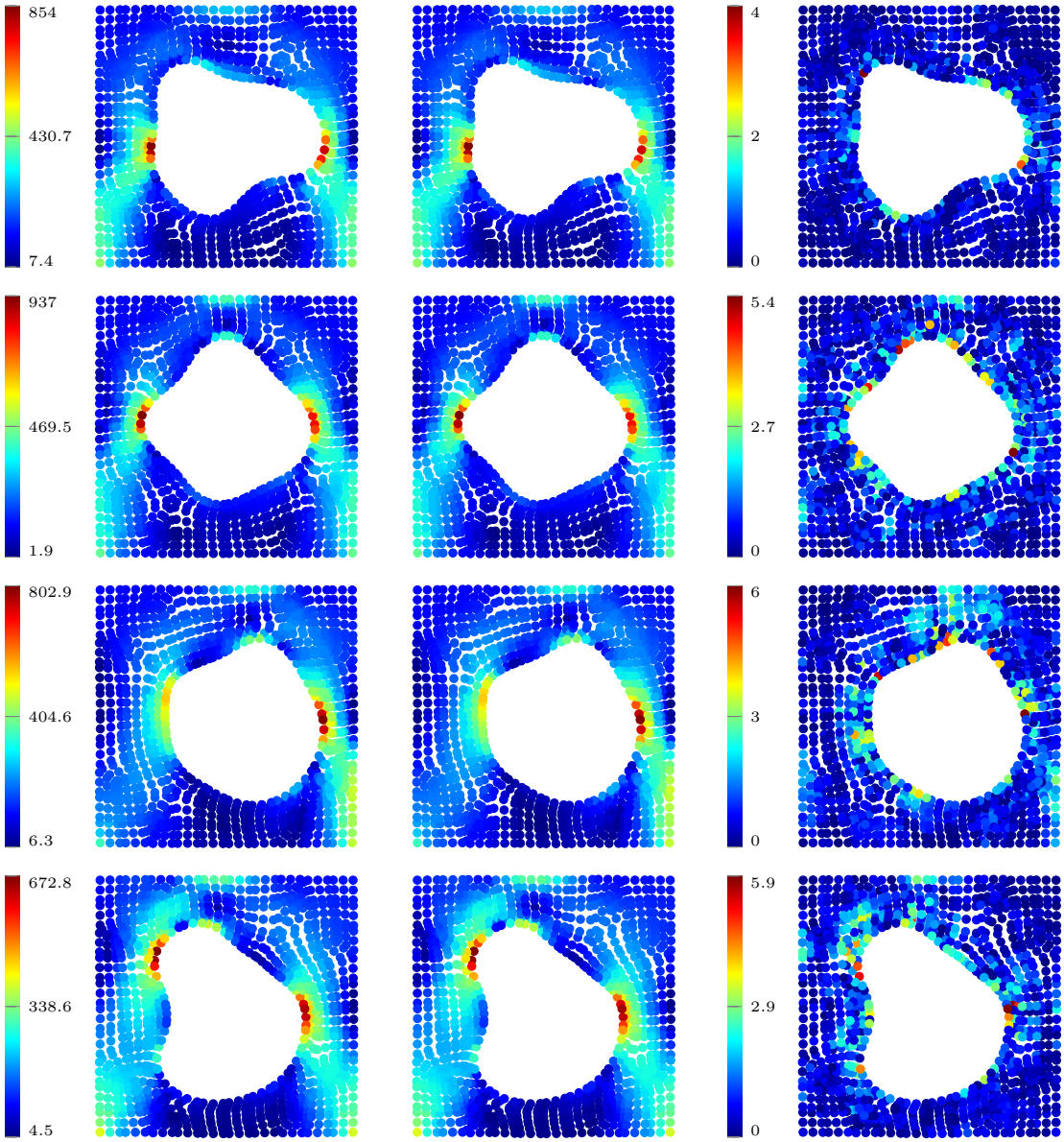

Figure 10. Stress field for elasticity. Left: reference. Middle: prediction by PiT. Right: absolute error.

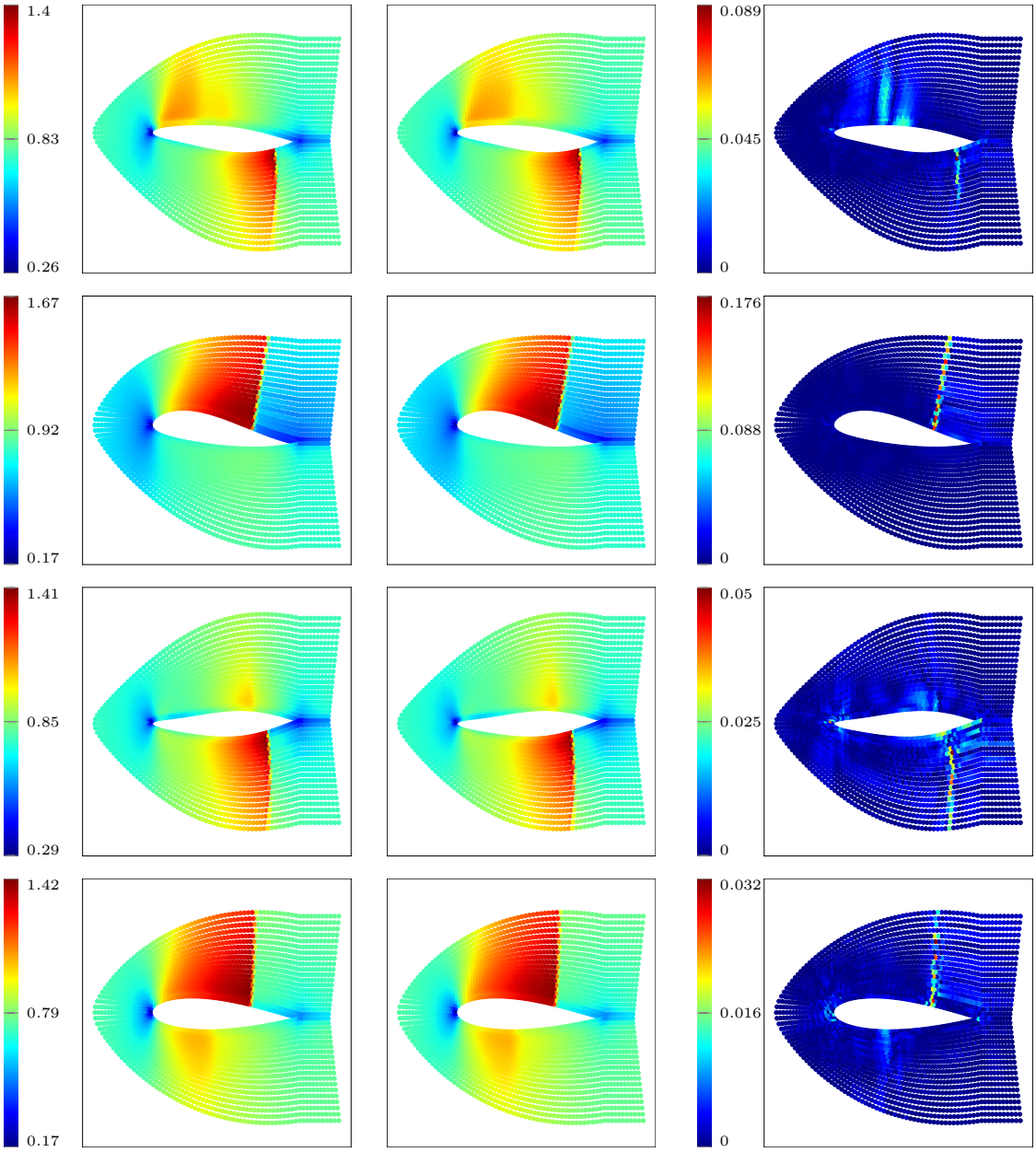

Figure 11. Fluid Mach numbers for NACA. Left: reference. Middle: prediction by PiT. Right: absolute error.